Patrik Asén

Analysis of the Flow Around a Cruise Ferry Hull by the Means of Computational Fluid Dynamics

School of Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 28.10.2014

Thesis supervisor:

Professor Timo Siikonen

Thesis advisor:

Tommi Mikkola, D.Sc. (Tech.)

Aalto University
School of Engineering

AALTO UNIVERSITY
SCHOOL OF ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Patrik Asén

Title: Analysis of the Flow Around a Cruise Ferry Hull by the Means of Computational Fluid Dynamics

Date: 28.10.2014 Language: English Number of pages:9+85

Department of Applied Mechanics

Major: Technical Mechanics Code: K3006

Supervisor: Professor Timo Siikonen

Advisor: Tommi Mikkola, D.Sc. (Tech.)

In this thesis computational fluid dynamics (CFD) is used to determine the drag of a cruise ferry in a double hull, as well as a free-surface case. The computations are done with OpenFOAM-software. Both cases are computed with five different grid spacings. The results obtained are compared to findings from towing tank experiments, as well as to each other in order to distinguish between the differences of the two cases.

The computations in OpenFOAM are based on the finite volume method. Turbulence is modelled with the STT k-ω method by Menter. The free surface is treated with the Volume of Fluid (VOF) method. The computations are done on a set of geometrically similar grids with the number of cells between $0.14 - 0.44$ million for the double hull cases, and between $0.6 - 2.2$ million for the free-surface cases. All the cases are run with the same Reynolds number of 20 million. The ship hull is considered to advance at a constant velocity in deep and calm water. The flow is considered incompressible.

The results are presented in a non-dimensional form. The drag is presented as the drag coefficient and it is compared to experimental towing tank results. The shear stress distributions as well as the pressure distributions for the double hull case are compared to those from the free-surface case.

The results obtained for the total drag are satisfactory when compared to earlier findings. The results were also accurate enough to distinguish clear differences between the double hull and free-surface cases. However, the grid refinement was not completely successful, as the solution did not converge on the finer grids. The reason behind this is not completely understood.

Keywords: CFD, OpenFOAM, Hydrodynamics, Turbulence Modelling, Free Surface Flow

AALTO-YLIOPISTO
INSINÖÖRITIETEIDEN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Patrik Asén

Työn nimi: Risteilylautan rungon virtauksen analysointi laskennallisen virtausmekaniikan keinoin

Päivämäärä: 28.10.2014 Kieli: Englanti Sivumäärä:9+85

Sovelletun mekaniikan laitos

Pääaine: Teknillinen mekaniikka Koodi: K3006

Valvoja: Professori Timo Siikonen

Ohjaaja: Tekniikan tohtori Tommi Mikkola

Tässä diplomityössä ratkaistaan laskennallista virtausmekaniikkaa käyttäen risteilylautan vastus sekä tuplarunkotapauksessa että vapaan nestepinnan kanssa. Laskenta on suoritettu avoimen lähdekoodin OpenFOAM-ohjelmistolla. Laskennassa on kummallekkin tilanteelle käytetty viittä eri hilatiheyttä. Saatuja tuloksia on verrattu aikaisempiin löytöihin. Tuloksia on myös verrattu keskenään, jotta eroja virtaustilanteiden välillä löydettäisiin.

Laskenta OpenFOAM-ohjelmassa perustuu kontrollitilavuusmenetelmään. Turbulenssimalleista käytettiin Menterin STT k-ω mallia. Vapaa nestepinta käsiteltiin VOF-nestetilavuusmallilla. Laskennat on suoritettu geometrisesti samanlaisilla hiloilla joiden koppimäärät ovat välillä $0.14 - 0.44$ miljoonaa tuplarungolle ja $0.6 - 2.2$ miljoonaa vapaalle nestepinnalle. Kaikkissa simuloinnessa pidettiin Reynoldsin lukuna 20 miljoonaa. Laivarungon oletetaan etenevän vakionopeudella syvässä ja tyynessä vedessä. Virtaus on luonteeltaan puristumatonta.

Tulokset on esitetty dimensiottomassa muodossa. Vastus on esitetty vastuskertoimina ja niitä on verrattu hinausaltaalta saatuihin koetuloksiin. Tuplarungon ja vapaan nestepinnan antamia leikkausjännityksiä ja painejakaumia rungolla on verrattu toisiinsa.

Tuloksia voidaan pitää kokonaisvastuksen kannalta tyydyttävinä, kun niitä verrataan aikaisempiin tuloksiin. Tulokset ovat myös riittävän tarkkoja, jotta selviä eroja tuplarungon ja vapaan nestepinnan tilanteiden välillä pystytään havaitsemaan. Hilatihennystä ei kuitenkaan saatu suoritettua loppuun, sillä suuremmilla hilatiheyksillä laskenta ei konvergoinut. Syytä tälle ilmiölle ei onnistuttu löytämään.

Avainsanat: CFD, OpenFOAM, hydrodynamiikka, turbulenssin mallinnus, vapaan nestepinnan virtaus

Acknowledgements

First and foremost, I would like to express my very great appreciation to my instructor D.Sc. Tommi Mikkola for giving me the opportunity to work for the Group of Ship Hydrodynamics and to embark on this master's thesis. He has dedicated numerous hours of his time to work with me on this project and has been a mentor and an insightful source of knowledge. He seemingly has an infinite amount of patience and has never left any questions of mine unanswered.

I am grateful to Professor Timo Siikonen for supervising this project. In addition, I would like to thank Professor Siikonen for being my teacher throughout my whole Master's programme.

The work was supported by the Innovations and Networks programme of the Finnish Metals and Engineering Competence Cluster, FIMECC. This financial support is greatly appreciated. I would also like to thank STX Finland for providing the cruise ferry geometry and the Finnish IT Center for Science (CSC) for providing the CPU time needed for this project. In addition, assistance provided by Mr. Esko Järvinen from CSC was greatly appreciated.

I would also like to extend my thanks to my colleague M.Sc. Pablo Esquivel de Pablo for not only guiding me through a multitude of difficulties I encountered during this project, but also for the fruitful discussions we have had.

Finally, I would like to thank all the Finnish taxpayers, who have made it possible for me to study for free.

Otaniemi, 28.10.2014

Patrik E. W. Asén

Contents

Symbols and abbreviations

Roman Symbols

A	constant
a	matrix coefficient
B	constant for the log-law layer
B_{WL}	maximum beam at waterline
$\mathbf{b}$	body force
C	Courant number
C_f	skin friction coefficient
C_D	closure coefficient for the SST $k-\omega$ turbulence model
C_d	draf coefficient
C_T, C_F, C_R	coefficients for total, friction and residual resistances
$\mathbf{d}$	vector between P and N
F_n	Froude number
F_1, F_2	1st and 2nd blending functions for the SST $k-\omega$ turbulence model
$\mathbf{H}$	matrix term
h	grid spacing parameter
$\mathbf{i}, \mathbf{j}, \mathbf{k}$	unit vectors in x, y and z directions
k	turbulent kinetic energy
k	hull form factor
L_{PP}	length between perpendiculars
L_{WL}	length at waterline
L_r	reference length
l_e	average size of energy containing eddies
m	mass
N	neighbouring cell centroid
P	cell centroid
P_k	production of turbulent kinetic energy
p	pressure
Q	arbitrary flow variable
$\mathbf{q}$	source term
R_c	transverse curvature
R_F	friction resistance
R_T	total resistance
R_R	residual resistance
S	invariant measure of strain rate
S	wetted surface
$\mathbf{S}$	surface area vector
S_{ij}	mean strain rate tensor
T	integration interval
T	draft
t	time
$\mathbf{U}$	instantaneous velocity in vector notation

U	reference velocity
U_r	artificial velocity
U_∞	free stream velocity
u_i	instantaneous velocity in tensor notation
u_τ	friction velocity
u^+	dimensionless velocity
V	volume
x_i	spatial coordinate in the ith direction
y	wall distance
y^+	dimensionless wall distance

Greek symbols

α	under-relaxation factor
α_1, α_2	closure coefficients for the SST $k-\omega$ turbulence model
$\beta_1, \beta_2, \beta^*$	closure coefficients for the SST $k-\omega$ turbulence model
$\delta_{i,j}$	Kronecker delta
δ^*	displacement thickness
δ_2	displacement thickness in case of transverse curvature
ϵ	dissipation per unit mass
η	Kolmogorov length scale
Θ	momentum thickness
κ	von Kármán constant
λ	eigenvalue
μ	dynamic viscosity
μ_t	dynamic eddy viscosity
ν	kinematic viscosity
ν_t	kinematic eddy viscosity
ρ	density
$\sigma_{k1}, \sigma_{k2}, \sigma_{\omega 2}, \sigma_{\omega 2}$	closure coefficients for the SST $k-\omega$ turbulence model
τ	condition number
τ_w	wall shear stress
ϕ	volume fraction
ω	specific dissipation rate
ω	weight factor

Superscripts

n	time level
T	transpose
$'$	fluctuating around the mean value
$\bar{}$	mean value
$\circ$	old time level

Subscripts

f	value on the cell face
m	model
N	neighbouring cells
P	owner cell
s	ship

Operators

∇	gradient
$\nabla\cdot$	divergence
$\frac{D}{Dt}$	material derivative

Abbreviations

CD	Central Differencing Scheme
CFD	Computational Fluid Mechanics
CV	Control Volume
DNS	Direct Numerical Simulation
FVM	Finite Volume Method
GAMG	Geometric Algebraic Multigrid Method
ITTC	International Towing Tank Conference
LES	Large Eddy Simulation
LTS	Local Time Stepping
PBiCG	Preconditioned Biconjugate Gradient
PDE	Partial Differential Equation
PISO	Pressure-Implicit with Splitting of Operator
RANS	Reynold Averaged Navier-Stokes Equations
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
SST	Shear Stress Transport
UD	Upwind Differencing
VOF	The Volume of Fluid method

1 Introduction

Shipbuilding has been practised since pre-historical times, and advances in the field have been essential in the rise of our modern society. Today safety demands together with increasing ecological awareness lead to evermore stricter demands for new ships. For the shipbuilding industry to meet these requirements, new and/or enhanced methods have to be applied. In general, two different approaches can be taken, namely experimental and mathematical.

While experimental methods have presumably been used throughout the history, the first mathematical approaches to ship hydrodynamics can be traced back to the 19th century [1]. Whereas experimental methods focus on measuring (most often) real model scale bahviour of a ship or part of it, mathematical approaches concentrate on modelling these.

The rapid growth of computer capacities during the past half-century has opened new horizons for mathematical approaches. One of these is the use of computational fluid dynamics (CFD). With CFD the governing equations of fluid flow are solved numerically, and a numerical solution for the whole flow field is attained as a result. As the flow around a ship hull is a highly complex problem, CFD demands a lot of computational effort and has thus been out of the reach of hydrodynamicists until lately. The development of CFD-tools has followed that of computers, more computational capacity has led to more complex simulations. Modern supercomputers can solve full scale problems with free surface flow and turbulence, a quantum leap from the mathematical methods used before the emergence of CFD.

However, completely calculating all the different scales of a flow problem, known as Direct Numerical Simulation (DNS), is still out of question even for the most powerful supercomputers. This is due to the wide range of length scales involved in turbulence. Thus, different methods have been developed where parts of the flow problem are modelled instead of being completely solved. Currently the most important such models are the different Reynolds-Avaraged Navier-Stokes Equations (RANSE) where the fluctuations of the flow are not solved accurately in the spatial dimension. Instead, only an averaged solution is attained. The RANSE solutions are important for academia as well as many industrial fields as they produce high quality results compared to their computational requirements.

Despite of the emergence of CFD, experimental methods are still today highly important, and results from towing tanks are essential in the ship design process. As the viscous forces and the free-surface effects cannot directly be scaled, the central question with model scale results is the scaling to full-scale. The International Towing Tank Conference (ITTC) has in 1957 determined a single correlation line [2] to be used in ship model testing. As the total resistance is a sum of several factors, and the flow around a ship hull exhibits complex behaviour, it is hard to separate the different sources of resistance. With the ITTC-57 friction line, the total resistance is divided into a friction term and a residual term. The friction resistance is estimated as a function of Reynolds number, while the residual resistance is, as the name suggests, the difference between the total resistance and the friction resistance.

Separation of these two resistance components as described above can be seen as

highly superficial. As ITTC-57 is essentially only a correlation line, it does not take into account different hull shapes at all. To better understand the different sources of resistance, results from a double hull case with no free-surface and a towing tank case with free-surface can be compared. Supposedly, the difference of these two values of resistance should be the resistance caused by the free-surface. However, this approach also has its flaws. The main problem being the fact that the flow for a double hull is substantially different from a free-surface case. The free-surface will deform and create a much different flow regime around the hull when compared to the double hull. It is simply not correct to just separate the friction forces from the free-surface forces, as these two are strongly coupled.

In this master's thesis, the flow around a cruise ferry is simulated with `OpenFOAM`-software. The aim of the work is to find the drag coefficient for a double hull and a free-surface case, and to distinguish differences between the flow regimes of the two cases. In case there are differences to be found, where are they and why do they exist? For such a study, CFD is an excellent tool, as detailed data about the flow can be extracted from the whole flow regime. At the same time, the suitability of `OpenFOAM`-software for future projects at the Aalto University Ship hydrodynamics group is also studied. Finally, this work aims to serve as an introduction to the `OpenFOAM`-software for future work in the field of ship hydrodynamics.

In the thesis, both a double hull case without a free-surface as well as a free-surface case are solved with `OpenFOAM`-software. The grids have been generated with the HEXPRESS™software, and post-processing has been done with ParaView. The simulations are done in a model scale of $1 : 22.713$. Turbulence is modelled with Menters SST $k - \omega$ model. The double hull cases were calculated with the `simpleFOAM` solver, while the free-surface cases were computed with the `LTSInterFOAM` solver. As `OpenFOAM` extensively uses its own terminology, all the jargon specific to the software has been denoted with the `typewriter` font.

For both the double hull and the free-surface cases, five different grid densities were used in order to complete a mesh refinement study. For a comparison of the flow regimes between the two cases to be meaningful, the meshes were kept as similar to each other as possible. However, some refinements to the free-surface case had to be added in order to capture the relevant phenomena accurately. The mesh refinement study could not be executed completely, as the finer mesh levels would not converge. The reason for this is outside the scope of this work.

As a whole, this work has been an iterative process. As the free-surface modelling turned out to be extremely cumbersome, emphasis was given to robustness of the solution. The mesh as well as the discretization methods and all other parameters associated with the simulations are not necessarily optimal from an accuracy standpoint, they have simply been iteratively found to be robust while providing an acceptable level of accuracy. As a result, a complete reasoning for all the methods used cannot be given, nor is there any guarantee that these methods could be extended to similar projects.

2 Background

2.1 The Ship Design Process

Ship design is a complex engineering decision-making process involving the integration of a multitude of different subsystems into a final solution. A design process is unique in the sense that it always has to meet specific requirements and cannot, therefore, be completed through a predefined scheme. Moreover, the process is bounded by both time and financial limits.

The process is commonly described by a design spiral presented in Fig. 1. This spiral was introduced by Evans in 1959 [3] and represents an iterative and step-wise procedure that produces results which may be acceptable but not optimal. After completing all the steps on the spiral, the results of an iteration round are analysed and modified, whereafter the modified results are re-analyzed until the requirements are satisfied. This iterative nature of ship design is due to the complexity of the process. As the problem can not be described by a single set of equations which can be solved directly, an iterative approach has to be taken.

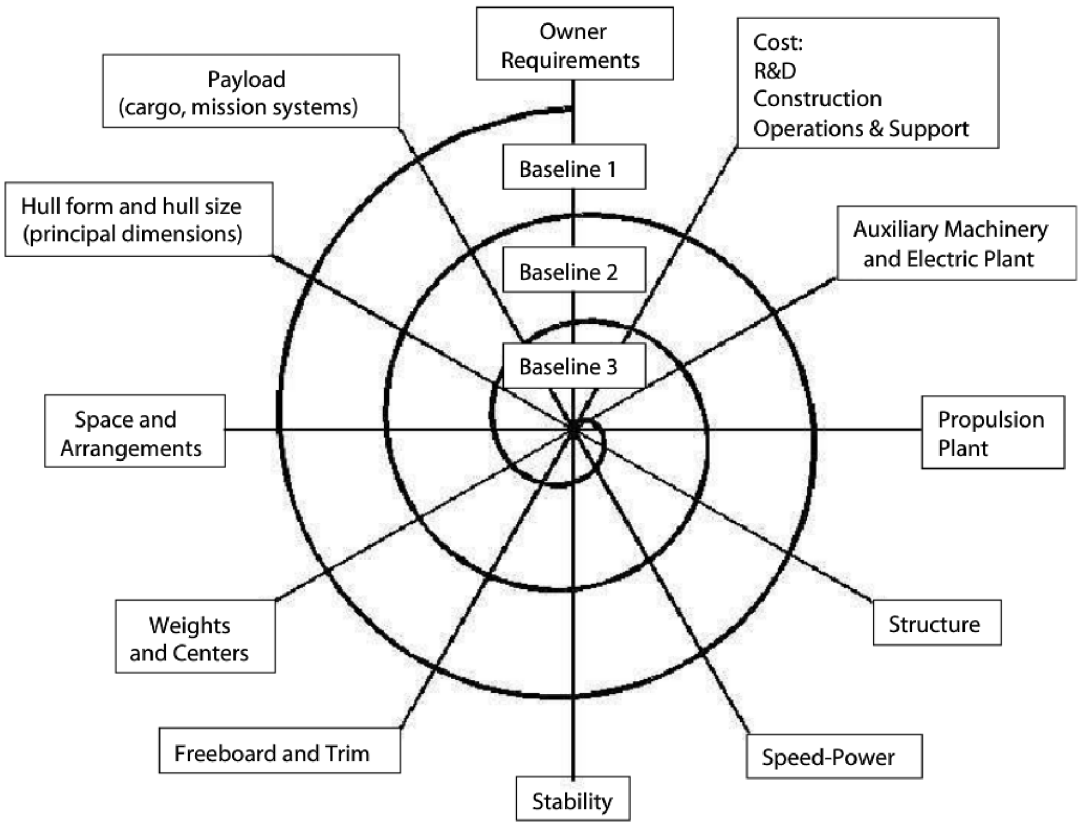

Figure 1: The ship design spiral. [4]

In reality, the design process is not as sequential as the design spiral depicts. The early stages of a design process might be highly unpredictable, the naval architects

might in fact jump between the different spots on the spiral. Once a baseline concept has been established in sufficient detail, the design work in the principal disciplines, such as propulsion and weight estimations, can proceed in parallel. For each of these disciplines, a series of tasks must be completed. As the tasks are completed, the results for the discipline can be shared across the design team. These results might very well require reworks of tasks already completed. [4]

Modern ship design is most often done with the help of computer-based tools. In fact, these tools are basically a requirement in order to complete the design of modern ships. Recent tools, known as the product model programs, aim to include all the different aspects for the engineering, design, construction and maintenance of a ship in a single package. Such tools provide a single source of updated and consistent information to all involved in the design and production processes, something which will be even more important in the future.

As the costs of shipbuilding increase, as do the requirements of new ships, computer software will probably play an integral role in the future of ship design. The ship designers of future cannot necessarily afford to follow the classical ship design spiral where the key design parameters are fixed in the early stages of the design. This approach makes late-state design fixes costly and often leads to sub-optimal ship designs. Instead, the problem is approached by investing more resources up-front. A versatile and robust design software is necessary for this up-front approach to be possible. [5]

2.2 Computational Fluid Dynamics

Fluid flow is a complex problem which can mathematically be described through partial differential equations. These equations are complicated and cannot be analytically solved in general cases. The field of Computational Fluid Dynamics (CFD) focuses on obtaining numerical solutions to these equations. To obtain a numerical solution, the equations governing the fluid flow have to first be discretized, i.e., the partial differential equations are approximated by a set of algebraic equations. These algebraic equations can then be solved on a computer. The approximations are applied to small domains in space and time, so the solution provides results at discrete locations in space and time as well. The typical workflow of a CFD problem consists of three parts, namely pre-processing, simulation and post-processing.

The discretization of the real life problem is the first step of preprocessing. Here, the real life geometry is subdivided into small subdomains. The most often used discretization method in CFD is the Finite Volume Method, where the solution domain is subdivided into a finite number of small control volumes. After the computational domain has been created, the problem has to be initialized by choosing which physical phenomena to model and how these will be modelled. The phenomena governing ship flow will be presented in Sec. 3 and onwards, while the computational methods for modelling these will be given in Chapt. 5.

The simulation part includes the actual computations of the discretized problem. The computational requirements vary depending on the problem size and the level of complexity of both the geometry and physics. Most often the computations are

so demanding that supercomputers are used to attain a numerical solution. In such cases, the case is solved in parallel with multiple processors working on their respective part of the problem. Thus, the case has to be decomposed into multiple smaller ones before the simulation can be done.

In the post-processing part the results of the simulation are analysed. Here the raw numbers crunched from computations are visualized in a more illustrative format. However, this stage is not limited only to visualization. Most importantly, the results have to be confirmed to be reliable. In addition, post-processing often includes calculation of new derived quantities, based on the original results. Most often post-processing is the last step of a loop, as the relationship between different stages of the workflow are iterative, i.e., when a problem is detected or a non-satisfactory result is obtained, the whole process will start again by revising the choices made at the pre-processing stage.

CFD has become an important tool in the field of fluid dynamics. As the computational capacity of computers continue to grow, bigger and more complicated problems can be solved. The benefits of CFD compared to traditional fluid flow experiments are many, with the major ones being listed below [6]

- Relatively low cost, no need to set-up and run physical experiments.
- Speed, set-up for a CFD problem is faster than for a physical one. In addition, changes to the original design can be made quickly.
- Comprehensive data can be extracted from CFD, whereas a physical test case can only provide data from a limited number of locations. In addition, there is no testing apparatus interacting with the flow.
- Greater control of the set-up of the experiment. Conditions which would be difficult or impossible to achieve in a physical test can be easily created in CFD.

Based on the list above, CFD appears to be too good to be true. In fact, the above advantages for CFD are conditional on being able to solve the governing equations of the fluid problem accurately. This is by no means a trivial task. As shall be seen in this work, finding an accurate solution for a complex flow problem is currently out of reach of modern supercomputers. Thus, the results obtained from CFD are always approximations. The three major sources of inaccuracies are [6]

- *Models or idealizations* are used to make a numerical solution feasible. In practise, turbulence is always modelled, other examples include combustion and multiphase flow. Even if a model would be solved accurately, the solution itself is not a correct representation of reality. Thus, models introduce inaccuracies to the results.
- *Discretization errors* arise when the problem is discretized. This error can be reduced by using more accurate interpolations or by applying this to a finer region. These will increase the computational cost, hence a compromise between accuracy and cost is needed.

- *Iterative* methods are used to solve the discretized equations. These methods usually solve the discretized equation only to a limited accuracy, and an error is produced. Much like the case with discretization errors, even here a compromise between accuracy and cost is needed.

Error estimation is an important part of CFD and it could be seen as an own discipline. In this work, potential sources of errors are many, and they are mentioned whenever a model or method with error potential is introduced.

2.3 CFD in Ship Design

As mentioned at the start of this chapter, computers have become an integral part of the modern ship design process. This is also true for CFD, with the growing computational capacities at hand more complex applications for CFD have been made within naval architecture. The most important applications of CFD in ship hydrodynamics are [7]

- *Resistance* in calm water is the most often used application for CFD. In the early days of CFD, potential flow was used to get approximate solutions for the ship resistance. During the last two decades, viscosity and wave making have drifted into CFD applications.

- *Manoeuvring* considerations of ships have become more important with the grown safety demands. CFD methods have become important in this field, as model tests are expensive and time-consuming. Computational models have progressed during the last years, but the large scatter of results between simulations has kept CFD from becoming the preferred approach for manoeuvring problems.

- *Seakeeping* studies have been dominated by panel methods, which do give satisfactory results. Recently, more advanced CFD methods including turbulence modelling have been introduced in order to solve problems characterized by strong non-linearities.

- *Propeller flows* have been extensively studied by computational methods. Inviscid lifting-surface methods have been shown to yield results comparable to experiments. Viscous CFD methods can be applied to more complex propeller configurations. In the future it is expected that CFD will be able to solve problems involving the ship together with the rudder and propeller.

The value of CFD in ship hydrodynamics consists of the general benefits of CFD, presented in Sec. 2.2. Firstly, we take into consideration the time benefits achieved by CFD. As already mentioned, the shipbuilding industry works with evermore complicated project and tighter schedules. In some cases, delivery time is the key factor for getting a contract. CFD plays a special role in this context, as a numerical pre-optimization can save time-consuming iterations in model tests and thus reduce total development time. Early use also reduces development risks for new ships.

This is especially important for unconventional ships where design cannot be based on previous experience.

In addition to time, another important strength of CFD are cost benefits. Direct cost savings are somewhat limited, as shipyards most often do not rely on CFD alone, but also perform at least one model test. However, indirect cost savings, although difficult to quantify, are obvious. Firstly, the time savings will directly cut down on costs. In addition, the expensive modifications at late stages of ship design projects can be avoided as different designs can quickly be reviewed with CFD. Thus, CFD will lower the financial risk of the whole ship design project.

The third aspect associated with CFD is quality. Although towing tank tests are considered reliable and have been used for a long time to determine ship resistance, CFD provides insight in flow details not provided by experiments. This becomes important if the results from the towing tank shows problems or the shipowner is willing to pay extra for lower operating costs associated with a better hull. In such cases, CFD allows the investigation of the flow in much greater detail than experiments. CFD can indicate where and how a design could be improved, and it also allows for rapid optimization of hull designs.

3 The Governing Equations

The aim of CFD is to solve the basic equations governing fluid flow, namely the continuity equation and the Navier-Stokes equations. Together these equations constitute a closed system for the pressure p and the three velocity components in the x, y and z directions, i.e., u, v and w, respectively. Figure 2 presents the coordinate system used throughout this work. Here x is directed towards the fore, y to port side and z vertically upward.

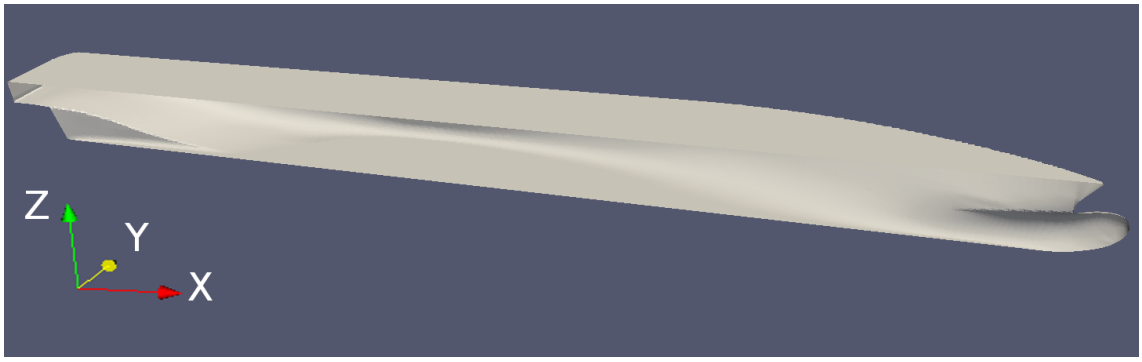

Figure 2: The coordinate system.

3.1 The Continuity Equation

The continuity equation describes the conservation of mass in a control volume, i.e., mass cannot disappear nor can there be any generation of mass. In differential form, the continuity equation is [8]

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0, \tag{1}$$

where ρ is fluid density, t is time and $\mathbf{U}$ is the velocity vector. For a steady state flow, Eq. (1) is reduced to

$$\nabla \cdot (\rho \mathbf{U}) = 0. \tag{2}$$

As all the fluids in this work will be treated as incompressible, the equation can be further simplified to its final form

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0. \tag{3}$$

3.2 The Navier-Stokes Equations

The equation of motion for a fluid, know as the Navier-Stokes equation, is an extension of Newton's second law to the motion of a fluid under the assumption that the

total stress acting on a particle is the sum of a pressure term F_p , a viscous term F_v and a body force term F_b.

The pressure term can be derived from Fig. 3. The force per mass m acting on the body in the x-direction will be

$$\frac{dF_{px}}{dm} = -\frac{1}{\rho}\frac{\partial p}{\partial x} \tag{4}$$

In a similar way terms can be developed for the y and z-directions, thus pressure in the final Navier-Stokes equation will be written as $-\nabla p$.

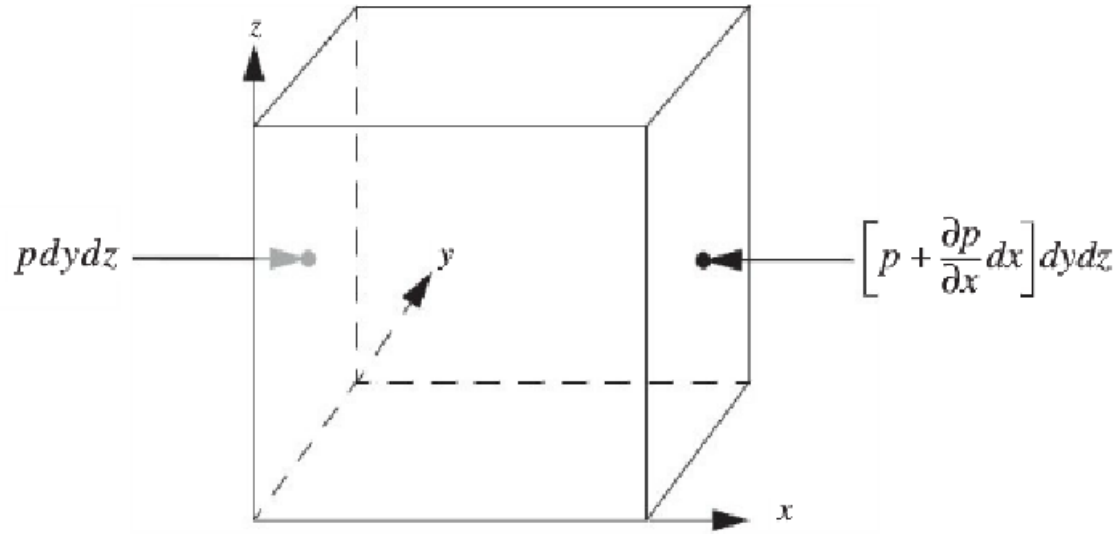

Figure 3: Pressure acting on the faces of a fluid element.

The viscous term turns out to be considerably more complicated. As Fig. 4 shows, the term consists of stresses acting on the sides of a fluid particle both in the tangential as well as the normal directions.The stresses σ are identified by two indices, the first one denotes the surface on which it acts while the second one its own direction. From Fig. 4 the total viscous force acting on the fluid element in the x-direction can be written as

$$\frac{dF_{vx}}{dm} = \frac{1}{\rho}\left[\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z}\right]. \tag{5}$$

To determine the stresses, a constitutive relation between the viscous stresses and the rate of strain is needed. This work will only treat Newtonian fluids which obey the following conditions

$$\sigma_{ij} = \mu S_{ij}, \tag{6}$$

where μ is the dynamic viscosity and S_{ij} is the rate of strain tensor, defined as

$$S_{ij} = \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}. \tag{7}$$

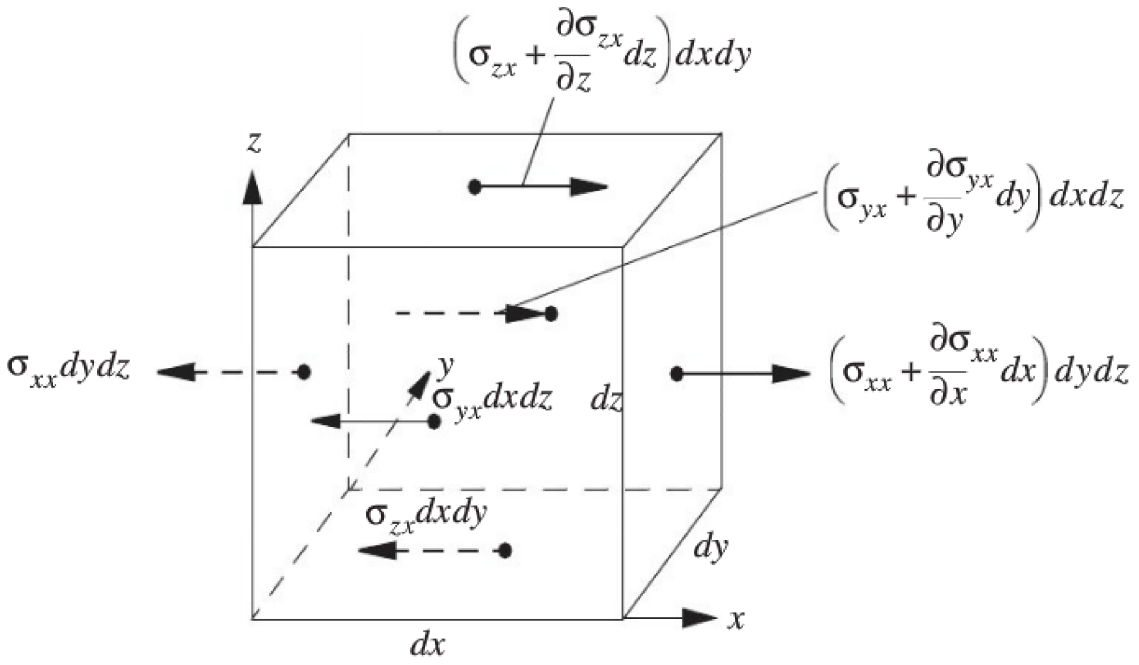

Figure 4: Viscous stresses acting in the x-direction on a fluid element.

Through these conditions and Eq. (3), Eq. (5) can be rewritten as

$$\frac{dF_{vx}}{dm} = \frac{\mu}{\rho}\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right]. \tag{8}$$

As was the case for the pressure, a similar treatment can be given to the y and z-directions as well, giving the total force due to viscosity as $\mu\nabla^2\mathbf{U}$.

The only body force considered in this work will be the gravity, working in the negative z-direction. Thus, the Navier-Stokes equation for an incompressible Newtonian fluid can now be written as

$$\rho\frac{D\mathbf{U}}{Dt} = -\nabla p + \mu\nabla^2\mathbf{U} + \rho\mathbf{g} \tag{9}$$

where $\frac{D\mathbf{U}}{Dt}$ denotes the material derivative, i.e., $\frac{D\mathbf{U}}{Dt} = \frac{\partial\mathbf{U}}{\partial t} + \mathbf{U}\cdot\nabla\mathbf{U}$ and $\mathbf{g}$ is gravity.

The governing equations consist of partial differential equations (PDEs) which are solvable only for a very limited number of simple cases. For engineering problems, numerical approaches have to be taken and these will be presented in Chapt. 5.

3.3 Turbulence

Whereas laminar flow displays regular and smooth flow paths, turbulent flow is characterized by a highly irregular and unsteady flow in both spatial and temporal dimensions. A laminar flow will become unstable and a transition to turbulence occurs when the Reynolds number reaches a critical value. The Reynolds number is

a ratio between the inertial forces and viscous ones, and is defined as

$$Re = \frac{\rho U L_r}{\mu} = \frac{U L_r}{\nu}, \tag{10}$$

where L_r is the reference length and μ and ν are the dynamic and kinematic viscosities, respectively. The value of the critical Reynolds number is always case specific, e.g. for a flat plate the transition to turbulence occurs at a Reynolds number close to $5 \cdot 10^5$. [8]

The difficulty to accurately predict turbulence lies in the wide range of different scales contained in a high Reynolds number flow. In 1922 Richardson [9] introduced a concept known as the energy cascade which explains the eddy motions of turbulence. According to this concept, the large eddies are unstable and break up, transferring energy to smaller eddies. The process continues and energy is transferred to successively smaller eddies, until at a sufficiently small size, the eddies become stable and dissipate. Important findings about the small scale eddies were made in 1941 by Kolmogorov [10], who found that these have a universal form uniquely determined by dissipation. In addition he found the Kolmogorov length scale η which defines the length at which, the eddies dissipate to heat as follows

$$\eta = \left(\frac{\nu^3}{\epsilon} \right) \tag{11}$$

where ϵ is dissipation of turbulent kinetic energy per unit mass. Assuming that the velocity field can be decomposed into two parts, the averaged velocity $\overline{u}$ and the turbulent fluctuation u', dissipation can be expressed as

$$\epsilon = \frac{A u'^3}{l_e} \tag{12}$$

where A is a a constant of the order of unity and l_e is an average size of the energy containing eddies. From Eq. (11) and (12), under the assumptions that $l_e \propto L_r$ and $u' \propto U$, it follows that

$$L_r/\eta \propto Re^{3/4}. \tag{13}$$

To resolve all the scales, the number of computational cells in every direction has to be of the same order as the ratio of L_r/η. [11]

For a a full scale simulation of a ship hull, where the Reynolds number can go up to 10^9, a grid of 10^{20} would be required in order to solve all the scales. This would lead to a computational demand far beyond the capacity of modern super computers. Thus, different strategies have been developed in order to predict turbulent motion.

3.3.1 Turbulence modelling

As turbulent flow occurs in nearly all practical applications, it has been the subject of numerous studies in the field of CFD. Although an exact definition of turbulence has eluded scientists, different methods have been developed to model it. As modern supercomputers do not provide enough computational capacity to directly solve

turbulent flows, different models for turbulence have been developed. The most widely used tools for turbulence modelling are as follows

1. Direct Numerical Simulation (DNS), no turbulence model is used, the governing equations are solved at all the scales.

2. Large Eddy Simulation (LES), the large scales of the flow are resolved, while the small scales are modelled.

3. Reynolds-averaged Navier-Stokes (RANS) equations are time-averaged equations for fluid motion. This averaging makes the model computationally friendly, but gives birth to terms known as the Reynolds stresses that must be modelled. Reynolds Stress Model (RSM) is a higher level turbulence model where the Reynolds stresses are not modelled but calculated. [6, 12]

3.3.2 Reynolds-averaged Navier-Stokes equations

Of the three models mentioned above, the RANS approach is the most widely used one for engineering applications. This approach is computationally more friendly since it only solves the mean quantities of the flow and thus avoids the small scale fluctuations of turbulence. By doing this, RANS offers a computationally feasible approach to turbulence modelling. Although RANS is widely used, its capabilities remain limited. The accuracy of a RANS solution often leaves something to be desired as it cannot describe large energy containing eddies. In addition, it can only produce time-averaged flow fields and cannot accurately predict important phenomena such as flow separation.

In order to elude the computationally expensive small scales of turbulence, any variable Q of a steady flow can be decomposed into a fluctuating part Q' and a time averaged part $\overline{Q}$

$$Q = Q' + \overline{Q}, \tag{14}$$

where the time-averaging is defined as

$$\overline{Q} = \lim_{T\to\infty} \frac{1}{T} \int_{t_0}^{t_0+T} Q \, \mathrm{d}t. \tag{15}$$

Here T is the integration interval, which must be large compared to the time scale of the fluctuations, but is not formally the limit $T \to \infty$.

Applying this decomposition to the velocity and pressure in the continuity Eq. (1) and Navier-Stokes Eq. (9), and then taking the time average, yields the RANS equations, named after Osbourne Reynolds who proposed the idea in the 19th century. For a constant density, both the fluctuating and the time averaged parts satisfy the continuity equation separately. However, this is not true for the non-linear Navier-Stokes equation which will now become

$$\rho \frac{D\overline{\mathbf{U}}}{Dt} + \rho \frac{\partial}{\partial x_j} \left(\overline{u_i' u_j'} \right) = -\nabla \overline{p} + \mu \nabla^2 \overline{\mathbf{U}} + \mathbf{b}. \tag{16}$$

Clearly, the equation is now further complicated by the new term $\overline{u_i' u_j'}$, known as the Reynolds-stress tensor. The presence of this tensor means that the equation is not closed, i.e., the number of unknowns is greater than the number of equations. The terms $\overline{u_i' u_j'}$ are not only related to the fluid properties, but also to the flow conditions. In essence, the procedure has introduced six new unknowns that can only be defined through unavailable knowledge of the turbulent structure. [8]

In order to attain closure to the problem, a turbulence model has to be introduced. Depending on the nature of the turbulence model, closure can be attained through the following three models:

1. Zero-equation models are the simplest of all turbulence models. They compute the Reynolds-stress tensor as the product of an eddy viscosity and the mean strain-rate tensor, through the Boussinesq eddy-viscosity approximation. Thus, the Reynolds stress tensor can be written as

$$\left(\overline{u_i' u_j'}\right) = \mu_t \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right) - \frac{2}{3} \rho \delta_{ij} k = \tau_{ij} \tag{17}$$

where μ_t is eddy viscosity, δ_{ij} the Kronecker delta and k turbulent kinetic energy , defined as

$$k = \frac{1}{2} \left(\overline{u_i' u_i'}\right). \tag{18}$$

The final term in Eq. (17) ensures that the equation remains correct when the two indices are set equal. The eddy viscosity is not a physical parameter, instead it depends on the flow conditions. In its simplest form, the eddy viscosity can be described through two flow parameters, namely the turbulence velocity $q = \sqrt{2k}$, and length scale L, giving the following expression

$$\mu_t = C_\mu \rho q L \tag{19}$$

where C_μ is a dimensionless constant. However, C_μ is not really a constant, in reality it is a ratio between the turbulent quantity to a a mean flow quantity. Thus, by treating it as a constant, additional inaccuracy is introduced to the model.

Through the Boussinesq approximation, the Reynolds stress tensor is replaced by only one unknown, namely the eddy viscosity. This is of course a drastic simplification, but the method is easy to implement and provides reasonably accurate results for many flows.

2. One-equation models extend on the Boussinesq approximaion by introducing a turbulence kinetic energy equation in order to attain closure to the RANS equations. This equation is defined as

$$\rho \frac{\partial k}{\partial t} + \rho u_j \frac{\partial k}{\partial x_j} = \tau_{ij} \frac{\partial u_j}{\partial x_j} - C_D \rho \frac{k^{3/2}}{l} + \frac{\partial}{\partial x_j} \left[(\mu + \mu_t / \sigma_k) \frac{\partial k}{\partial x_j} \right], \tag{20}$$

where C_D and σ_k are closure coefficients. There are a multitude of different one-equation models based on k. However, the one-equation models based on Eq. 20 are no longer being used. Especially more complicated flow with abrupt changes from wall-bounded to free shear flow, or separation, cannot be predicted accurately. Instead of k, modern one-equation models are based on some other variable. For example the Spalart-Allmaras model, which have been extensively used in aerodynamics, solves the transport equation for a viscosity-like variable [13].

3. Two-equation models are based on the Boussinesq approximation Eq. (17) and the equation for turbulent kinetic energy Eq. (20). In addition to the turbulence kinetic energy, the two-equation models contain a second parameter, most often the dissipation per turbulent kinetic energy ω, or dissipation per unit mass ϵ. The two-equation models have served as the foundation for much of the turbulence modelling done during the last decades. In this work, turbulence will be modelled by a two-equation model know as the SST $k-\omega$ model which will be presented in more detail later.

It is worthwhile to notice that all of these turbulence models are approximative, and do not apply universally to all turbulent flows. The general approach is to introduce a minimum amount of complexity while capturing the relevant physics of the problem.

3.3.3 Menter Shear Stress Transport Model

The Shear Stress Transport (SST) model is a two-equation eddy-viscosity model, introduced by Menter in 1994 [14]. The starting point for the development of the model was the need to accurately predict aeronautics flows with adverse pressure gradients and separation. As neither the $k-\omega$ nor the $k-\epsilon$ model can manage this alone, the principal idea of the SST model is to combine the two models in order to enhance their capabilities. The $k-\epsilon$ model is robust, but suffers from weaknesses in the modelling of the boundary layer [15]. Meanwhile the $k-\omega$ model is more accurate than the $k-\epsilon$ model in the near wall layers and is better for predicting flows with adverse pressure gradients, but is highly sensitive to the free-stream value of ω [16].

In order to achieve a more accurate model, Menter proposed a combination of the two. In his SST model, the $k-\omega$ formulation is used in the boundary layer, whereafter a switch to the $k-\epsilon$ formulation is made for the free-stream flow. Hence, the major problems associated with $k-\epsilon$ (inaccuracy at the boundary layer) and $k-\omega$ (free-stream dependency of ω) models are both solved. This zonal approach is based on a blending function, which ensures a proper selection of models without user interaction. This adds to the complexity of the SST model when compared to the original models, as the blending functions require the computation of the distance to the wall.

The most recent version of the SST $k-\omega$ model, with a limited number of modifications, was given by Menter in 2003 [17] and consists of the following formulas

$$\frac{\partial k}{\partial t}+\frac{\partial(u_i k)}{\partial x_i}=\frac{\partial}{\partial x_i}\left[\left(\nu+\frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_i}\right]+P_k-\beta^* k\omega, \tag{21}$$

$$\frac{\partial \omega}{\partial t}+\frac{\partial(u_i \omega)}{\partial x_i}=\frac{\partial}{\partial x_i}\left[\left(\nu+\frac{\nu_t}{\sigma_\omega}\right)\frac{\partial \omega}{\partial x_i}\right]+\alpha\frac{\tilde{P}_k}{\nu_t}-\beta\omega^2+2(1-F_1)\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{x_i}\frac{\partial \omega}{\partial x_i}, \tag{22}$$

where k is turbulent kinetic energy, ν kinematic viscosity, ν_t kinematic eddy viscosity, P_k production of turbulent kinetic energy, ω specific dissipation rate and $\sigma_k, \beta, \beta^*, \sigma_\omega, \sigma_{\omega 2}$ are closure coefficients. The blending function F_1 is defined as

$$F_1=\tanh\left\{\left\{\min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega y},\frac{500\nu}{y^2\omega}\right),\frac{4\rho\sigma_{\omega 2}k}{CD_{k\omega}y^2}\right]\right\}^4\right\}, \tag{23}$$

where

$$CD_{k\omega}=\max\left(2\rho\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i},10^{-10}\right), \tag{24}$$

and y is the distance to the nearest wall. The turbulent eddy viscosity is defined as

$$\nu_t=\frac{a_1 k}{\max\left(a_1\omega, SF_2\right)}, \tag{25}$$

where $S=\sqrt{2S_{ij}S_{ij}}$ is the invariant measure of strain and F_2 is the second blending function defined as

$$F_2=\tanh\left[\left[\max\left(\frac{2\sqrt{k}}{\beta^*\omega y},\frac{500\nu}{y^2\omega}\right)\right]^2\right]. \tag{26}$$

To prevent the build-up of turbulence in regions of stagnation, a production limiter is used

$$P_k=\mu_t\frac{\partial u_i}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)\rightarrow\tilde{P}_k=\min(P_k,10\beta^*\rho k\omega) \tag{27}$$

From Eq. (23) it is clear that F_1 becomes zero away from the surface and the $k-\epsilon$ model will be activated, while the $k-\omega$ model will be used at the boundary layer where F_1 becomes close to one. Each of the constants is a blend of an inner (1) and an outer (2) constant, blended via $A=A_1F_1+A_2(1-F_1)$, where A is an arbitrary constant. The constants for the SST model are presented in Table 2.

The SST model was originally designed for aeronautics applications, but has since become popular for a wide range of problems in both scientific and industrial applications [17]. In ship hydrodynamics, the SST model is among the most widely used turbulence models [18]. The models performance in flows involving separation makes it a valuable tool in ship hydrodynamics, as separation often occurs in the region of ship stern.

Table 2: The constants of SST model.

β^*	α_1	β_1	σ_{k1}	$\sigma_{\omega 1}$	α_2	β_2	σ_{k2}	$\sigma_{\omega 2}$
0.09	5/9	3/40	0.85	0.5	0.44	0.0828	1	0.856

3.3.4 Near-wall Treatment

The turbulent boundary layer is most often presented on a logarithmic scale, with the help of a dimensionless wall distance y^+, and a dimensionless velocity u^+, defined as

$$\begin{aligned} y^+ &= u_\tau y/\nu \\ u^+ &= u/u_\tau \end{aligned} \tag{28}$$

where $u_\tau = \sqrt{\tau_\omega/\rho}$ is the friction velocity and τ_w is the wall shear stress. The velocity profile of a flow near a wall can be divided into three parts:

1. The viscous sublayer is the layer closest to the wall, at a $y^+ < 5$. Here the velocity profile is linear, i.e., $u^+ = y^+$ and turbulence is damped out. The layer is instead dominated by viscous shear. The layer is very thin, for a flat plate around 500 times less than the entire boundary layer thickness.

2. Between $5 < y^+ < 30$ lies the overlap region. Here the velocity profile is a smooth merge between the (inner) linear and (outer) logarithmic ones.

3. The part of the boundary layer at $30 < y^+ < 300$ is known as the log-law layer. Here the velocity profile is logarithmic and can be expressed as

$$u^+ = \frac{1}{\kappa} \ln y^+ + B, \tag{29}$$

   where $\kappa = 0.41$ is the von Kármán constant and $B = 5.0$.

4. The outermost part of the boundary layer will commence at $y^+ > 300$. This region is known as the outer layer. Here, the boundary layer starts to merge with the free-stream flow. The velocity and turbulence fields approach their respective free-stream values exponentially fast [19]. [8, 21]

A schematic overview of the near-wall region is presented in Fig. 5.

Treating near-wall flows with CFD presents two difficulties. Firstly, most two-equation models (excluding the SST model) yield unsatisfactory results when integrated through the viscous sublayer. Secondly, integration through the viscous sublayer is computationally expensive as the first cell height has to be $y^+ = 1$ and a fine grid is required throughout the boundary layer in order to capture the physics of the problem accurately.

The problems described above can be circumvented through the introduction of wall functions. Here, the flow is not accurately computed with the actual no slip

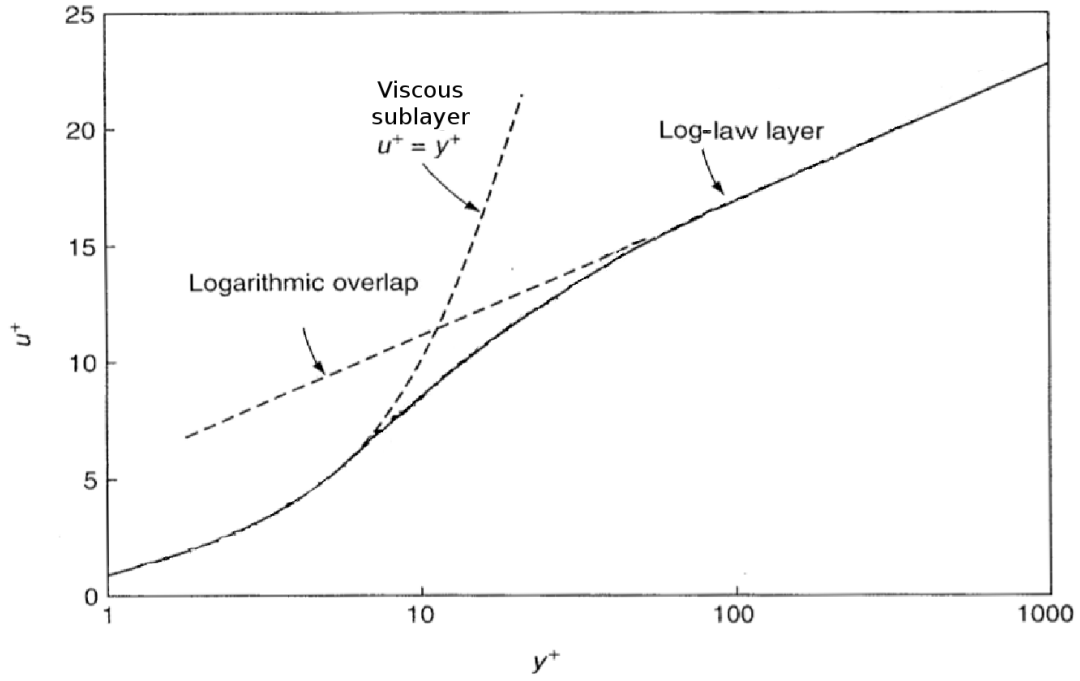

Figure 5: A schematic view of the near-wall region. Modified after [8]

boundary condition $U = 0$, instead the velocity profile is modeled. Due to the similar behaviour of different turbulent boundary layers, wall functions use the law of the wall as the constitutive relation between velocity and shear stress. This allows the use of a substantially coarser grid, with the height of the first cell $y^+ \approx 30$. Thus, major computational savings are achieved.

Due to the nature of the wall functions, the use of them will reduce the accuracy of the computation. As the velocity is not computed close to the wall, the physics in this region are not captured. This is especially a concern with detached flows, as the wall functions do not recognize this phenomenon. Additional concern is caused by the cell height requirements. As the y^+ values are not known a priori, the grid has to be generated iteratively. Finally, failure to set the y^+ within an appropriate range will lead to inaccurate results. For ship hydrodynamics studies, it has been shown that the viscous resistance attains an almost constant value when $30 < y^+ < 125$, and hereafter decreases rapidly [20].

3.4 The Volume of Fluid Method

The free-surface between water and air plays an integral role in ship hydrodynamics, and thus has to be treated carefully. In this work the volume of fluid (VOF) method is used for the treatment of the free-surface. VOF is a widely used surface capturing approach for flow problems involving multiple fluids, described by Hirt and Nichols

in 1981 [22]. The method is based on a fraction function which defines the fluids volume fraction in a cell. All fluids share a single set of momentum equations, while the volume fraction of each fluid is tracked through every cell. With the m:th fluid's volume fraction in a cell denoted as ϕ_m, the following three conditions are possible:

$$\begin{array}{cc} \phi_m = 0 & \text{the cell is empty of the fluid} \\ \phi_m = 1 & \text{the cell is full of the fluid} \\ 0 < \phi_m < 1 & \text{the cell contains a fluid interface} \end{array}$$

The normal direction of the fluid interface is found where the gradient of F_m is greatest. The tracking of the surface is attained by solving the continuity equation of the volume fraction for the m:th fluid in a system of n fluids

$$\frac{\partial \phi_m}{\partial t} + u_i \frac{\partial \phi_m}{\partial x_i} = 0 \tag{30}$$

with the following constraint

$$\sum_{m=1}^{n} \phi_m = 1, \tag{31}$$

i.e., the volume of the fluids is constant. For each cell, properties such as density and viscosity, are calculated by a volume fraction average of all liquids in the cell

$$\rho = \sum \rho_m \phi_m. \tag{32}$$

Finally, the properties from Eq. (32) are used to solve a single momentum equation through the domain, and the attained velocity field is shared among the fluids.

The VOF method is computationally friendly, as it introduces only one additional equation for each fluid and thus requires minimal storage. The method is also characterized by its capability of dealing with highly non-linear problems in which the free-surface experiences sharp topological changes. By using the VOF method, one also evades the use of complicated mesh deformation algorithms used by surface-tracking methods. Finally, VOF allows for flexible and simple grid generation. [18]

The major problem with VOF is the difficulty to discretize Eq. (30) without smearing the free-surface. This will be addressed in the section below. Additional problem with the method is that the free-surface is not defined sharply, instead it is distributed over a the height of a cell. Thus, in order to attain accurate results, local grid refinements have to be done. The refinement criterion is simple, cells with $0 < \phi < 1$ have to be refined. A method for this, known as the marker and micro-cell method, has been developed by Raad and his colleagues in 1997 [23].

3.4.1 Discretization Difficulties

The success of the VOF method depends heavily on the scheme used for advecting the ϕ field. The main difficulty arises from the need to treat the discrete interface as an averaged scalar value over a cell. This can be illustrated by considering the advection of a rectangular fluid region over a time interval δt with a Courant

number of 0.5 [24]. The upwind scheme gives the profile depicted in Fig. 6. The profile shows heavy smearing, because the volume fraction is treated as a standard scalar field rather than a discrete interface.

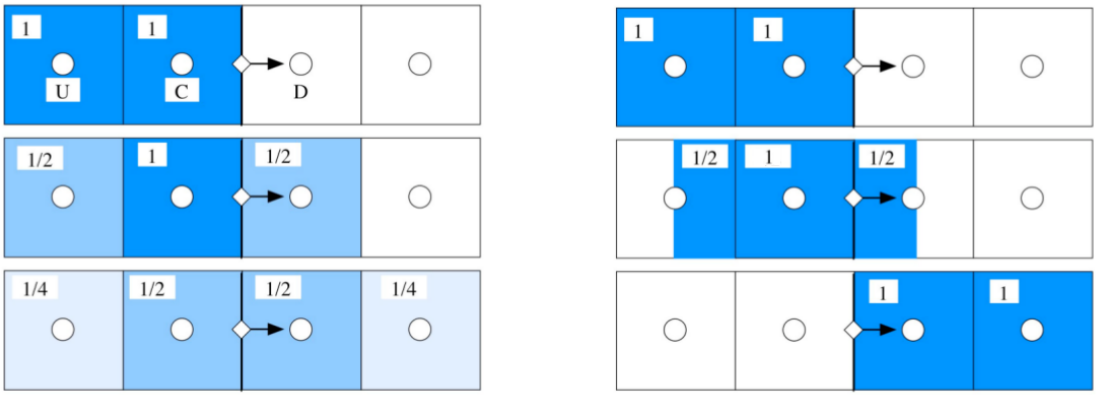

Figure 6: Advection of a fluid block at a Courant number of 0.5, using the upwind scheme (left) and the exact solution (right) [24].

A more appropriate treatment would be to use a downwind interpolation. However, another difficulty associated with first-order schemes is the false diffusion problem, which arises when the flow is not oriented along a grid line as depicted in Fig. 7. In practise, this phenomenon rules out the possibility of using simple first-order schemes with VOF.

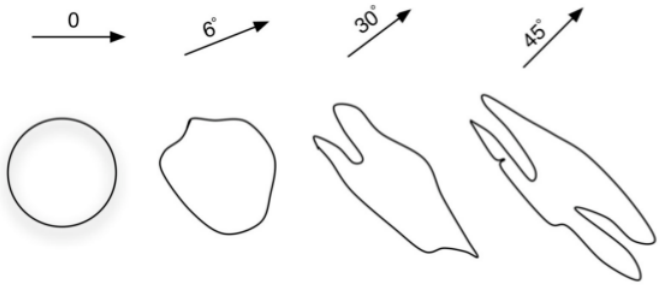

Figure 7: A shape of an initially round droplet after advection at four angles [24].

As the lower order schemes smear the interface and higher order methods are unstable and induce oscillations, it has been necessary to develop advection schemes that keep the interface sharp and produce monotonic profiles of the ϕ function. Over the years, a multitude of different schemes for VOF have been proposed by many researchers. In the original VOF-article by Hirt, a donor-acceptor scheme was employed. This scheme acts as a basis for the modern differencing schemes. For a scheme to be successful it has to be compressive by its nature, and in addition fulfil the boundedness criterion, i.e., the value of ϕ has to be between 0 and 1 [25].

However, there is no universal method which would be accurate in all cases. Thus, the VOF scheme should be chosen depending on the flow problem and previous experience from equivalent cases.

4 Viscous Flow Around a Ship Hull

In a flow around a ship hull, viscous forces are concentrated in the region near the hull and in the wake, i.e., regions with strong velocity gradients. Viscous effects outside of these regions can be neglected due to a lack of velocity gradients, making the viscous forces negligible. The surface of the hull fulfils a no-slip condition which states that the fluid particles on the surface do not have any velocity. Moving away from the surface, the velocity gradually increases within a small thickness until it reaches the value of the free stream flow. This area close to the hull is the boundary layer, it covers the whole surface and grows in thickness downstream.

The viscous effects are significant in the boundary layer, as it is dominated by strong velocity gradients. The boundary layer starts as laminar, but quickly becomes unstable. The instabilities grow until transition to a fully turbulent boundary layer occurs when a critical Reynolds number is exceeded. The value of the critical Reynolds number is always case specific, e.g. for a flat plate the transition to turbulence occurs depending on the surface roughness, at a Reynolds number between $5 \cdot 10^5 - 3 \cdot 10^6$ [8]. Regardless of the case, however, the innermost part of the boundary layer, known as the viscous sublayer, will stay essentially laminar.

4.1 The Boundary Layer of a Flat Plate

The flow over a flat plate will be used as a starting point for this boundary layer analysis. The flat plate is considered smooth, infinitely wide and aligned with the flow. A schematic view of a flat plate boundary layer is given in Fig. 8. As the pressure along a flat plate can be considered undisturbed, the behaviour of the boundary layer will be simple by its nature. The first solutions for the flat plate boundary layer were given by Blasius in 1908 [26], who was able to derive the boundary layer equations which greatly simplify the governing equations of fluid flow. Turbulent flow over a flat plate has since been extensively studied, and numerous explicit formulas have been proposed for the friction forces. These forces are often presented in a non-dimensional form through the skin friction coefficient

$$C_f = \frac{\tau_w}{\frac{1}{2}\rho U_\infty}, \tag{33}$$

where τ_w represents the wall shear stresses and U_∞ denotes the free stream velocity. When comparing local skin friction coefficients in this work, the following formula proposed by Kestin and Persen is used [8]

$$C_f = \frac{0.455}{\ln^2(0.06 \cdot Re_x)}. \tag{34}$$

The boundary layer thickness δ is the distance in y direction from the wall at which the boundary layer merges with the free stream. Conventionally the edge of the boundary layer is the point where the velocity equals 99% of the free stream velocity. As the boundary layer merges smoothly to the outer flow, the boundary

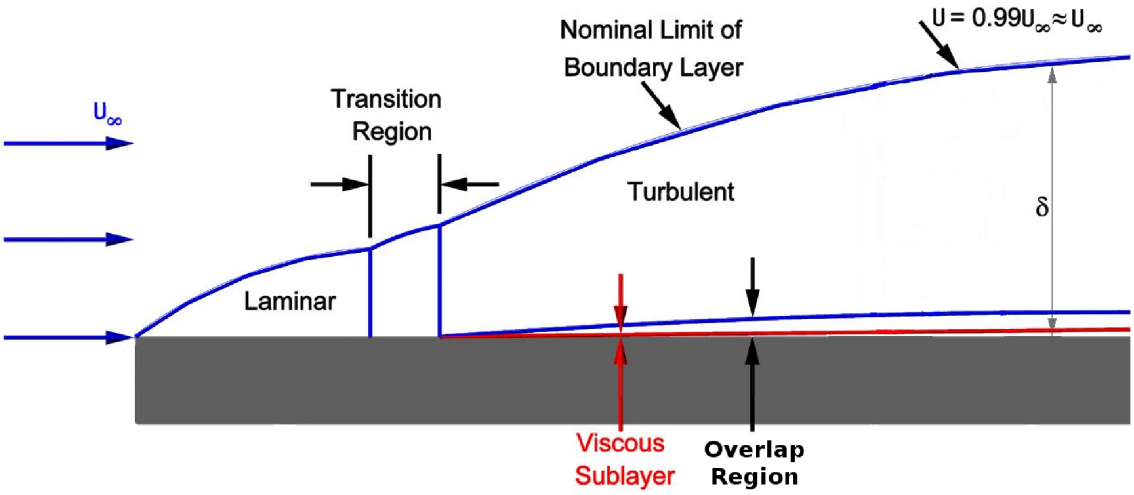

Figure 8: The boundary layer.

layer thickness cannot be defined directly. Instead, different boundary layer thicknesses can be defined, namely the displacement thickness δ^*, and the momentum thickness Θ, defined as

$$\delta^* = \int_0^\infty \left(1 - \frac{u}{U_\infty}\right) \mathrm{d}y \tag{35}$$

$$\Theta = \int_0^\infty \frac{u}{U_\infty}\left(1 - \frac{u}{U_\infty}\right) \mathrm{d}y. \tag{36}$$

The displacement thickness is the distance by which the surface would have to be moved in an inviscid fluid stream to give the same flow rate as occurs between the surface and the reference plane in a real fluid. In a similar fashion, the momentum thickness is defined as the distance by which the surface would have to be moved in an inviscid fluid stream to give the same momentum as occurs between the surface and the reference plane in a real fluid. [8]

The boundary layer behaviour described above is strictly for the case of a flat plate flow. As a ship hull is a more complex problem, the flat plate analogy does not give a complete scope of the viscous effects on a ship hull. However, the flat plate does give a fundamental basis which can be extended by adding new flow features. These features can roughly be divided in three categories, namely the two-dimensional, three-dimensional and axisymmetric effects of a fully three-dimensional body.

4.2 2D effects

For a flat plate flow, a fluid element moving in the boundary layer is only affected by viscous forces as the pressure is constant. This is in general not the case for a two dimensional body. Instead, the pressure is not constant and pressure gradients will appear. In regions where the pressure is diminishing along the body, the fluid element will be accelerated while the fluid element in regions of increasing pressure will be decelerated. Due to the negative pressure gradient, the velocity at the boundary layer edge is in general larger than for a flat plate, thereby increasing the friction on the body. This phenomenon is known as the form effect on the friction.

Pressure gradients play an integral role in the separation phenomenon. Separation occurs when the increase in pressure exerts a decelerating force on the fluid elements close to the surface. If this force is great enough, the longitudinal motion of the fluid elements will stop, forcing the streamlines to leave the surface and instead creating a zone of eddies and vortices. This separation region is characterized by very small axial mean velocity, but large velocity fluctuations. Separation will cause drastic changes to the flow and pressure fields, and increase the pressure drag of the object.

4.3 3D effects

For a three dimensional body, the streamlines close to the surface will diverge out from the stagnation point at the front. These streamlines will converge at the stern stagnation point and will be the most spread at the section with the largest diameter. This will have an effect on the boundary layer development. Due to continuity, the boundary layer growth has to reduce at regions where the streamlines diverge, while the opposite is true for regions where the streamlines converge. Thus, the boundary layer thickness will grow slower at the fore and faster at the tail regions. Converging streamlines might also lead to separation know as the vortex sheet separation. Here converging streamlines near the surface will force the flow to leave the surface and the boundary layer is swept out. Two different boundary layers will then meet, and strong vortices will appear in the intermediate layer.

In addition to the longitudinal pressure gradients in a two dimensional case, a three dimensional case will also introduce pressure gradients in the lateral direction, i.e. in a direction parallel to the surface but at a right angle to the flow. This pressure gradient will bend the flow resulting in a cross-flow development. The boundary layer thickness δ_2 has the following correlation to the flat plate [27]

$$\delta_2 = \frac{\delta_{flatplate}}{\sqrt{1 + \frac{\delta_2}{3 \cdot R_c}}}, \tag{37}$$

where R_c is the transverse curvature. As this boundary layer will be thinner than that for a flat plate, the shear stresses as well as the friction resistance increase.

4.4 Ship Resistance

Ship resistance is a complex issue comprising a number of different elements. The total resistance of a ship can be divided in different manners, here the division is done following [28]. The total resistance can be decomposed in two major components, namely the wave resistance and the viscous resistance. This decomposition is presented in Fig. 9 below.

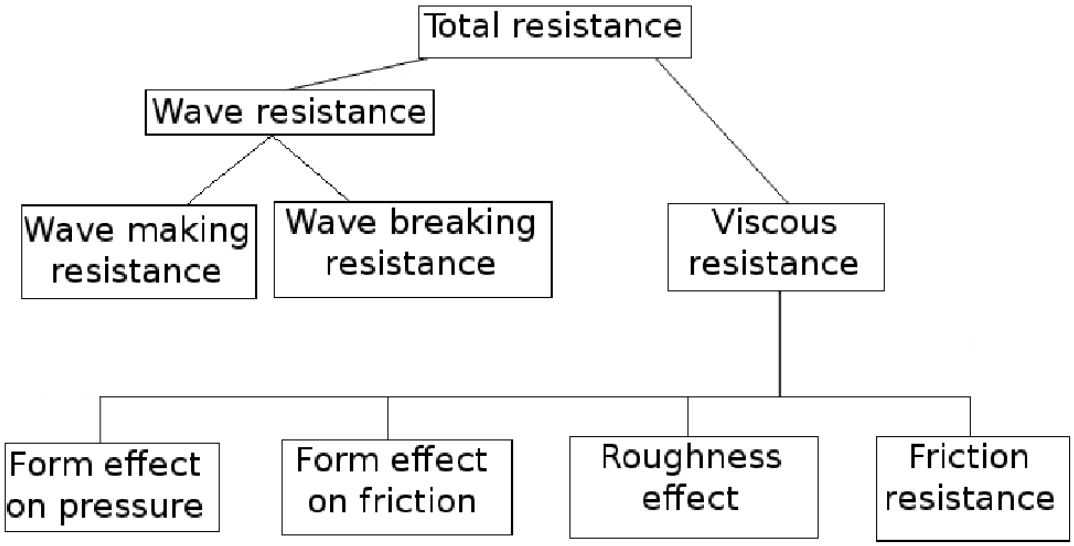

Figure 9: The decomposition of ship resistance

The wave resistance consist of two parts, the wave making resistance and the wave breaking resistance. As the ship moves along the free surface, water particles are removed from their equilibrium position, thus creating waves and giving rise to the wave making resistance. When the disturbances are large enough, the waves may break down into eddies. The energy removed from the wave system through this process is found in the wake of the ship and the corresponding resistance component is known as the wave breaking resistance. The remaining wave energy is propagated away from the ship and creates a wave pattern. This phenomenon gives rise to the wave making resistance. Wave breaking is outside the scope of this work and will thus not be covered in the following chapters.

The viscous resistance can be divided into four different parts: friction resistance, roughness effect and form effect on both friction and pressure. The latter two components are, as their names suggest, caused by the 3D shape of the ship hull. The form effect on pressure is caused by the pressure imbalance between the forebody and the afterbody. As the boundary layer develops along the surface of the ship, the streamlines will be displaced outward at the stern. Therefore, the pressure at the stern end of the ship will be reduced and the integrated pressure forces will not cancel.

The second form effect, i.e., the form effect on friction arises from the fact that the flow approaching the ship has to go around the hull. Thus, the velocity of the water close to the ship hull (but outside of the boundary layer) is different from that of the undisturbed flow ahead of the ship. The flow velocity is reduced at the bow and stern of the ship but over the main part of the hull the velocity is increased.

The friction resistance, also known as the skin friction, arises purely from the tangential forces between the ship hull and the water. A boundary layer with large velocity gradients is created close to the hull surface. This skin friction will be effected by the surface roughness in case the roughness exceeds a certain limit. For ship models, the surface roughness is negligible and is thus excluded from this work. Another neglected factor associated with the skin friction is the fact that a pressure loss will be induced by the skin friction and the water temperature at the hull will rise. This phenomenon could be modelled by solving the energy equation, but has not been included into this work.

4.5 Wave Making

Surface waves are created by local disturbances, such as the bow or other parts of a ship. A local disturbance will generate a continuous set of wave components which will propagate in various directions at angles between $-90° < \theta < 90°$. Waves with small values of θ are known as transverse waves, while waves with large angles are known as diverging waves. The waves will generate a fan-shaped pattern and will thus not fill the entire area behind the disturbance. This is due to the fact that the wave energy travels with the group velocity of the wave, which is only half of the phase velocity. Thus, the energy leaving the point of disturbance will not stay with the wave crest, but lag behind, which will ultimately lead to wave crest dying out. [28]

4.5.1 The Kelvin Wave Pattern

The phenomenon described above is the reason for a typical wave pattern created by a ship. Fig. 10 illustrates the features of a ship wave pattern, known as the Kelvin wave pattern [1]. The wave pattern consists of transverse and diverging waves, all contained within an wedge-shaped region, known as the Kelvin wedge, with a half-angle of 19.5° to the longitudinal axis.

4.5.2 Interference Effects

The wave system generated by a real ship is more complex than that described by the Kelvin wave pattern. A ship is not a single point disturbance, but has a number of wave generating points, such as the bow and shoulders. All of these will generate their own wave systems with separate transverse and diverging components, contained in their respective Kelvin wedges. The different wave systems created will overlap and interfere with each other.

In 1931, Wigley presented the interference between the different wave systems and the effect on the wave resistance [29]. As the surface waves are generated by

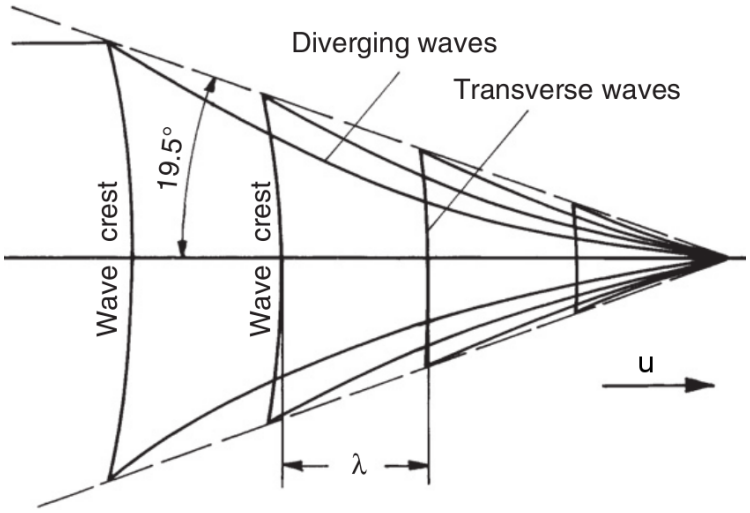

Figure 10: The Kelvin wave pattern. [7]

pressure disturbances, the most pronounced wave systems are created by the high pressure at the bow and stern as well as the low pressures at the shoulders. The latter two will produce waves starting with a trough, while the first two will generate waves starting with a crest. Altogether, the wave profile along the hull will contain the following five contributions [28]

1. The near field disturbance known as the Bernoulli wave.
2. The bow wave system, starting with a crest.
3. The fore shoulder wave, starting with a trough.
4. The aft shoulder wave, also starting with a trough.
5. The stern wave system, starting with a crest.

4.5.3 Wave Resistance

A body travelling on the surface will generate a wave pattern, and there will be a pressure distribution on the body. The resultant net longitudinal force is the wave-making resistance. This resistance must be of such a magnitude that the energy expended in moving against it equals to the energy needed to maintain the wave system. Although the foundations of theoretical methods for determining a ship wave resistance were laid by Mitchell in 1898 [30], the wave resistance cannot be calculated by simple design formulas. An important non-dimensional quantity when considering wave resistance is the Froude number, defined as

$$Fn = \frac{U}{\sqrt{gL_r}}. \tag{38}$$

The Froude number describes the ratio between the ship velocity and the gravitational wave velocity.

As the wave energy is proportional to the square of the wave amplitude, so is the wave resistance. Again, the interference of wave components plays an important role: constructive interference where wave systems amplify each other will lead to a large wave resistance, whereas destructive interference where wave systems cancel each other corresponds to low wave resistance. The interference depends on the Froude number, which will cause an oscillatory variation of the wave resistance with Fn. The Froude number at which a certain interference will take place depends strongly on the hull. It determines where the significant shoulder wave systems occur, what the type and location of the stern wave system is and what wave components prevail.

4.6 Similarity

Most experiments or simulations in ship hydrodynamics are carried out in model scale. In order for the model to accurately describe the real problem, similarity requirements for the wave resistance (Fn) as well as the viscous resistance (Re) have to be met. However, this is a dilemma as both of the requirements cannot be satisfied simultaneously.

In practise this has been solved by Froude [31] by satisfying the Froude number requirement and then correcting the errors on viscous resistance caused by the wrong Reynolds number. During the years, different approaches have been used, such as the ones proposed by Hughes [32] and Schoenherr [33]. In 1957 the 8th International Towing Tank Conference (ITTC) proposed a new standard to be used, namely the ITTC-57 friction line [2].

4.6.1 ITTC-57 Friction Line

The ITTC-57 friction line was formally accepted as a standard method in 1957 and is still widely used today. The central idea of the method is to use the Reynolds number to scale the friction from the model scale. The starting point for the method is to decompose the total resistance R_T into the friction resistance R_F and the residual resistance R_R

$$R_T = R_F + R_R, \tag{39}$$

and then to express these as non-dimensional coefficients

$$C_i = \frac{R_i}{\frac{1}{2}\rho V^2 S} \tag{40}$$

where S is the wetted surface. The friction resistance coefficient is estimated for the model and for the ship by the ITTC-57 friction line

$$C_{Fi} = \frac{0.075}{(\log Re_i - 2)^2}, \tag{41}$$

where $i = m$ for model and $i = s$ for ship. The measured model resistance R_{Tm} is made non-dimensional by

$$C_{Tm} = \frac{R_{Tm}}{\frac{1}{2}\rho_m V_m^2 S_m}. \tag{42}$$

The residual resistance coefficient C_R is the same for the ship and the model

$$C_R = C_{Tm} - C_{Fm}, \tag{43}$$

and the total resistance coefficient for the ship will be

$$C_{Ts} = C_{Fs} + C_R + C_a, \tag{44}$$

where C_a is the correlation coefficient, specific for the basin. An often used approach is to introduce a hull form factor k to describe the difference between the actual resistance and that given by Eq. 41 as follows

$$C_T = C_F + C_{PV} = (1 + k)C_{ITTC-57}, \tag{45}$$

where C_{PV} is the viscous pressure resistance coefficient.

5 Materials and Methods

5.1 The OpenFOAM Software

`OpenFOAM` (Open Source Field Operation and Manipulation) is a general purpose open-source CFD code. `OpenFOAM` is mainly a C++ library , used to create executables. These executables can either be solvers, that are designed to solve a specific problem in continuum mechanics, or utilities, that are designed to manipulate data. `OpenFOAM` contains a wide range of built-in solvers and utilities including turbulence modelling and multiphase flows. Currently the C++ library includes over 80 solvers and 170 utilities, all of which can be extended by the user to cover specific problems.

The `OpenFOAM` C++ library consists of not only the CFD solver itself, but a pre- and post-processing environment as well. Basically the idea of `OpenFOAM` is that the entire CFD process, from grid generation to post-processing, could be done through the same C++ library, thus ensuring consistent data handling across the process. However, ParaView which is the post processing software provided with the regular `OpenFOAM` package is a stand-alone software and not an actual part of `OpenFOAM` library. The overall structure of `OpenFOAM` is presented in Fig. 11. [34]

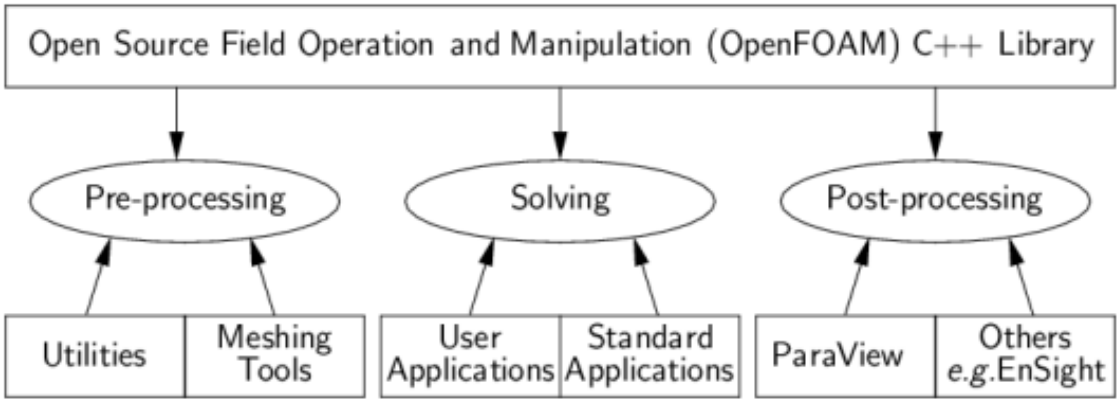

Figure 11: Overview of the `OpenFOAM` structure. [34]

As `OpenFOAM` is an open source project, it offers the user complete freedom to customize existing executable and add new ones. In addition, as `OpenFOAM` does not require any license, it can be executed in parallel without any licensing costs. This is essential for complex problems, such as ship hydrodynamics, where parallelization is required in order to achieve reasonable execution times for the simulations.

`OpenFOAM` is purely text based, and relies on a strict directory structure for each case. The structure of an `OpenFOAM` case is presented in Fig. 12, consisting of the following parts:

- The parameters associated with the solution procedure, such as the discretization schemes, are located in the *system* folder.

- The *constant* folder includes the constant properties of the problem, such as fluid properties. In addition, it contains the *polyMesh* folder where the geometry of the case is located.
- *Time* directories contain the data for each time step. The simulation is initialized from the 0 directory, i.e., this is where the boundary condition are defined at.

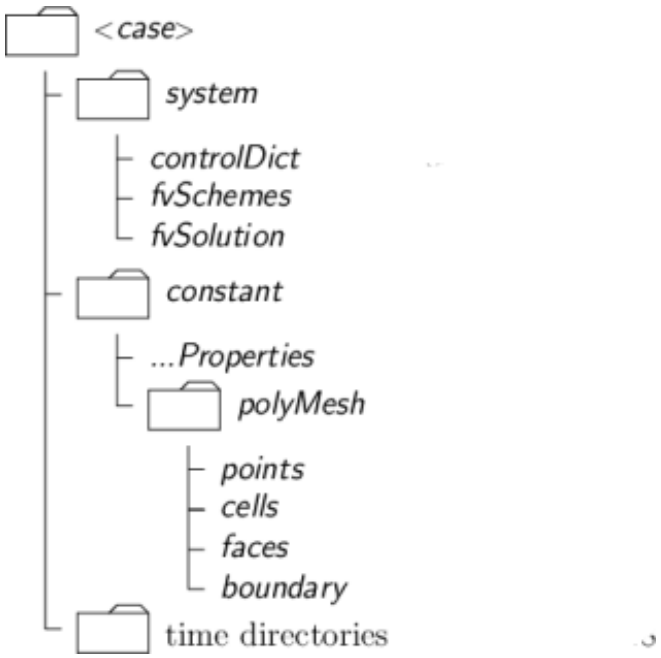

Figure 12: Overview of the case directory structure in `OpenFOAM` . [34]

All the free-surface cases were computed remotely at the Taito supercluster provided by CSC - IT Center for Science Ltd. Each case was computed in parallel with $16 - 32$ Intel Sandy Bridge 2.6 GHz processors, using `OpenFOAM` version 2.2.x.

5.2 Numerical Methods

The governing equations presented in Sec. 3 are all non-linear partial differential equations with continuous solutions. Analytical solutions to these equations are limited to only a small number of very simple cases. Thus, in order to solve these equations, numerical methods have to be introduced. In practise, as the equations are solved with computers, the equations will have to be brought into an algebraic form and additionally linearized. As a result, a system of algebraic linear equations is obtained and this set of equations is then solved.

In order for the continuous equations to be approximated by a set of discrete algebraic equations, the governing equations have to be formulated for discrete points in both space and time. There are different methods for doing this, with the most

important once being the finite volume, finite difference and finite element methods. Of these three, the finite volume method is the most often used approach in the field of computational fluid dynamics. Thus, the remaining of this chapter will focus on methods associated with the finite volume method.

5.2.1 The Finite Volume Approach

The finite volume method (FVM) is a numerical method for solving partial differential equations. As the name suggest, the FVM is based on subdividing the solution domain into a finite number of small control volumes (CVs). These CVs are contiguous, i.e., they completely fill the domain and do not overlap one another. The governing equations are then integrated over each cell. The cell notation used in `OpenFOAM` is depicted in Fig. 13. The notation of `OpenFOAM` will be used throughout the following subsections.

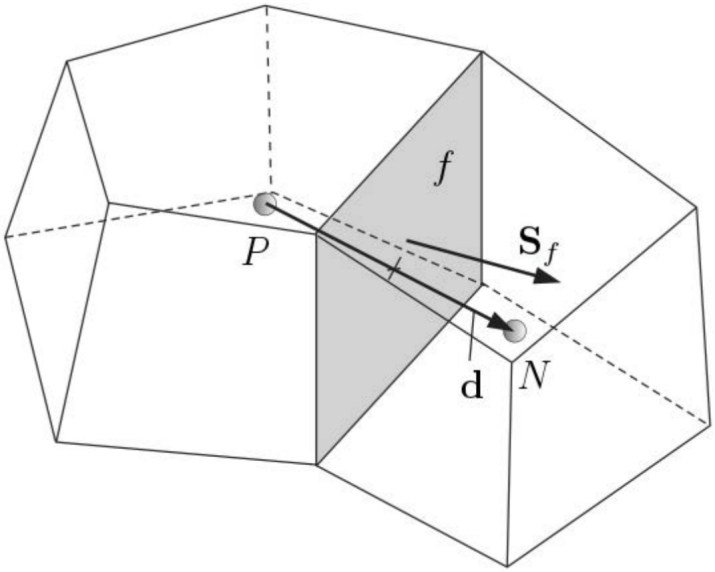

Figure 13: The notation used in finite volume discretization. [35]

Variables are stored at the cell centroids P and N. Each cell is bounded by faces f, and each face is owned by one adjacent cell while the other cell is called the neighboring cell. Each face has an area of $|\mathbf{S}_f|$ and a unit normal vector $\mathbf{n}$ pointing towards the neighbor. A surface area vector can thus be defined as $\mathbf{S}_f = |\mathbf{S}_f|\,\mathbf{n}$. The vector $\mathbf{d}$ in Fig. 13 points from the centre of the owner cell towards the centre of the neighbor cell $\mathbf{d} = \overline{PN}$. Finally, the volume of the cell is denoted as V_P. `OpenFOAM` does not restrict the shape of the cells, nor the alignment of the faces. This is known as an arbitrarily unstructured mesh, and it allows for flexible mesh generation in cases involving complex geometries. [35]

5.2.2 Equation Discretization

The FVM utilizes the integral forms of the governing equations presented in Sec. 3. The basic concept here is to convert the partial differential equations to a set of algebraic equations. A general conservation equation for a variable ϕ can be written as

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{U}\phi) = \nabla \cdot (\mu \nabla \phi) + \mathbf{q}_\phi. \tag{46}$$

In the FVM, the above equation is integrated over the volume of a CV as follows

$$\overbrace{\int_V \frac{\partial \phi}{\partial t}\, \mathrm{d}V}^{\text{time derivative}} + \overbrace{\int_V \nabla \cdot (\mathbf{U}\phi)\, \mathrm{d}V}^{\text{convection}} = \overbrace{\int_V \nabla \cdot (\mu \nabla \phi)\, \mathrm{d}V}^{\text{diffusion}} + \overbrace{\int_V \mathbf{q}_\phi\, \mathrm{d}V}^{\text{source}}. \tag{47}$$

Of the four terms above, the time derivative and the source term can be integrated straightforward, by multiplying with the cell volume V_P. The time derivative can be approximated in different ways. This will be treated in Sec. 5.2.5.

On the other hand, the treatment of the convection as well as the diffusion terms is not as straightforward. The volume integrals are converted to integrals over the cell surface S bounding the volume, using Gauss theorem

$$\int_V \nabla \star \phi\, \mathrm{d}V = \int_S \phi \star\, \mathrm{d}\mathbf{S}, \tag{48}$$

where ϕ represents any tensor field, $\mathbf{S}$ is the surface vector and $\star$ represents either the gradient $\nabla\phi$, divergence $\nabla \cdot \phi$ or curl $\nabla \times \phi$. By applying the above to the convective term, the volume integral will translate into a surface integral

$$\int_V \nabla \cdot (\mathbf{U}\phi)\, \mathrm{d}V = \int_S \phi \mathbf{U} \cdot \mathrm{d}\mathbf{S}. \tag{49}$$

Further, by approximating the surface integral through a sum over discrete surfaces, the convection term can be rewritten as

$$\int_V \nabla \cdot (\mathbf{U}\phi)\, \mathrm{d}V = \int_S \phi \mathbf{U} \cdot \mathrm{d}\mathbf{S} \approx \sum_f \phi \mathbf{U} \cdot \mathbf{S}_f. \tag{50}$$

In a similar manner, the diffusion term can be rewritten through Gauss's theorem and then approximated through a sum over discrete surfaces as follows

$$\int_V \nabla \cdot (\mu \nabla \phi)\, \mathrm{d}V = \int_S (\mu \nabla \phi) \cdot \mathrm{d}\mathbf{S} \approx \sum_f (\mu \nabla \phi) \cdot \mathbf{S}_f. \tag{51}$$

by combining the results from above, the discrete transport equation can now be written as

$$V_P \frac{\partial \phi}{\partial t} + \sum_f \phi \mathbf{U} \cdot \mathbf{S}_f = \sum_f (\mu \nabla \phi) \cdot \mathbf{S}_f + V_P \mathbf{q}_\phi. \tag{52}$$

The variables $\mathbf{U}, \alpha, \phi, \nabla\phi$ are usually expressed in cell centres, but as the above equation indicates, these variables have to be expressed on the cell faces f in the discrete formulation. Thus, a need for interpolation arises. A brief introduction to different interpolation methods will be given in the following section.

5.2.3 Interpolation

The basic upwind and linear differencing schemes will be introduced in this chapter, with the latter one being treated first. In central differencing (CD) the value of ϕ at the CVs face center is interpolated by weighting the two adjacent cell center values by their distances to the face

$$\phi_f \approx \omega\phi_P + (1-\omega)\phi_N, \tag{53}$$

where the weight factor ω is defined as the ratio of the neighboring cell center distance to the face $\overline{fN}$ and the owner cell center $\overline{PN}$

$$\omega = \overline{fN}/\overline{PN}, \tag{54}$$

and the notation follows that of Fig. 13. By using the Taylor series expansion, the CD scheme can be shown to be of second-order accuracy, i.e., the leading term of the truncation error is proportional to the square of the grid spacing. Thus, CD is the simplest second-order accurate method. However, as is the case with all higher order approximations, this method may produce oscillatory solutions. The oscillations are unbounded, and can at worst lead to unstable computations. [6]

Another straightforward method to attain the values at a face center is the upwind differencing (UD) scheme. Depending on the flow direction, either a backward- or forward-difference is used. The variable ϕ is approximated as

$$\phi_f = \begin{cases} \phi_P, & \mathbf{U}_f \cdot \mathbf{S}_f \geq 0 \\ \phi_N, & \mathbf{U}_f \cdot \mathbf{S}_f < 0. \end{cases} \tag{55}$$

This approximation unconditionally satisfies the boundedness criterion and will thus not produce any oscillations. However, it is only first-order accurate. The truncation error contains a first-order term which is diffusive by its nature and thus smooths out any sharp peaks in the values of ϕ. In addition, the first-order error is also greater in magnitude than one of the second-order. Thus, the UD scheme is robust by its nature, but it cannot be used to obtain results of satisfactory accuracy. [6]

To circumvent the obvious shortcomings of the schemes presented above, a multitude of different schemes and other approaches have been proposed. These include, but are not limited to, blending of different schemes, using limiters to avoid oscillations as well as using complex and/or higher order schemes. In this work, the convective terms, with the exception of the free-surface equation, have been interpolated with the second-order upwind scheme `linearUpwind`.

5.2.4 Pressure-Velocity Coupling

The coupling of velocity and pressure is one of the central problems of CFD. As can be seen from the governing equations, pressure does not have its own equation in incompressible flow. The pressure gradient is a part of the source term in the momentum equation, but there is no obvious way of obtaining it. To get around this difficulty, the continuity equation can be used to correct the pressures iteratively until both the continuity and momentum equations are satisfied. This approach was introduced by Harlow and Fromm in 1965. In the 1970s Patankar developed the famous and often used SIMPLE algorithm [36]. Since then, a multitude of different derivatives of the method have been proposed, but they all have their roots in the SIMPLE method. The `LTSInterFoam` uses the `PIMPLE` algorithm which is a merge between the SIMPLE and PISO algorithms. Thus, a short introduction to these two will be given below, following the notation used by Jasak [37], one of the main contributors behind `OpenFOAM`.

In order to attain an equation for pressure, the Navier-Stokes equations are written in a discretized form. The convective term Eq. (50) can be re-written as

$$\sum_f \mathbf{U} \cdot \mathbf{S}_f \mathbf{U} = a_P \mathbf{U}_P + \sum_N a_n \mathbf{U}_N, \tag{56}$$

where a_P and a_N are functions of $\mathbf{U}$. As the fluxes above should satisfy the continuity equation, Eq. (3) and (9) have to be solved together. This will result in a large non-linear system. As a complex non-linear system is computationally heavy, the equation above should be linearized. This implies that an existing velocity field which satisfies Eq. (3) is used to calculate a_P and a_N. This is especially true for transient flows, whereas a steady-state calculation is not affected by the linearization.

In order to derive the pressure equation, Eq. (9) will first be written in a semi-discretized form

$$a_P \mathbf{U}_P = \mathbf{H}(\mathbf{U}) - \nabla p, \tag{57}$$

where the $\mathbf{H}(\mathbf{U})$ term consists of two parts, namely a transport part consisting of matrix coefficients for all neighbours multiplied by the corresponding velocities, and a source part including all other source terms than the pressure gradient:

$$\mathbf{H}(\mathbf{U}) = -\sum_N a_N \mathbf{U}_N + \frac{U^\circ}{\Delta t}. \tag{58}$$

$\mathbf{U}$ can now be expressed through Eq. (57) as

$$\mathbf{U}_P = \frac{\mathbf{H}(\mathbf{U})}{a_P} - \frac{1}{a_P} \nabla p. \tag{59}$$

Further, the face velocities can be interpolated from the above equation as

$$\mathbf{U} = \left(\frac{\mathbf{H}(\mathbf{U})}{a_P} \right)_f - \left(\frac{1}{a_P} \right)_f (\nabla p)_f \,. \tag{60}$$

When the above equation is substituted into the discretized continuity equation

$$\nabla \cdot \mathbf{U} = \sum_f \mathbf{U} \cdot \mathbf{S}_f = 0, \tag{61}$$

one attains

$$\nabla \cdot \left(\frac{1}{a_P} \nabla p \right) = \nabla \cdot \left(\frac{\mathbf{H}(\mathbf{U})}{a_P} \right) = \sum_f \mathbf{S} \cdot \left(\frac{\mathbf{H}(\mathbf{U})}{a_P} \right)_f . \tag{62}$$

Finally, the discretized forms for Eq. (3) and (9) can now be written as

$$a_P \mathbf{U}_P = \mathbf{H}(\mathbf{U}) - \sum_f \mathbf{S}(p)_f, \tag{63}$$

$$\sum_f \mathbf{S} \left[\left(\frac{1}{a_P} \right)_f (\nabla p)_f \right] = \sum_f \mathbf{S} \left(\frac{\mathbf{H}(\mathbf{U})}{a_P} \right)_f . \tag{64}$$

The face flux is calculated from

$$\mathbf{S}\mathbf{U}_f = \mathbf{S} \left[\left(\frac{\mathbf{H}(\mathbf{U}}{a_p} \right)_f - \left(\frac{1}{a_P} \right)_f (\nabla p)_f \right] . \tag{65}$$

There is a clear inter-equation coupling between the two equations above, and this requires special treatment. Two such methods, namely the SIMPLE and PISO algorithms, will be presented next.

The SIMPLE algorithm is used to solve steady-state problems iteratively. For such a problem, it is not necessary to fully resolve the linear pressure-velocity coupling, as the changes between consecutive time-steps is no longer small. The algorithm is design to take advantage of this, as follows

- The momentum equation is solved to attain an approximation for the velocity field. The pressure distribution from the previous iteration is used to calculate the pressure gradient. This is known as the momentum predictor stage. The equation is under-relaxed with the velocity under-relaxation factor α_U.

- The pressure equation is constructed and solved using the predicted velocities. The solution of the pressure equation gives the first estimate of the new pressure field.

- The conservative fluxes which are consistent with the new pressure field are calculated from Eq. (65). Now, in order to attain a better approximation for the pressure field, the pressure equation should be calculated again. However, as this is not necessary for a steady-state problem, the new conservative fluxes are only used to recalculate the coefficients in $\mathbf{H}(\mathbf{U})$. The pressure solution is then under-relaxed in order to take into account the error caused by the velocity

$$p^{new} = p^{old} + \alpha_p \left(p^p - p^{old} \right), \tag{66}$$

where p^{new} is the approximation of the pressure field to be used in the next momentum predictor, p^{old} is the pressure field used in the momentum predictor, p^p is the solution of the pressure equation and α_p is the pressure under-relaxation factor. In this work, values of $\alpha_p = 0.3$ and $\alpha_U = 0.7$ were used.

For transient problems the PISO algorithm was originally proposed by Issa in [38]. The method follows the same path as the SIMPLE algorithm described above. However, an explicit velocity correction is done after the conservative fluxes for the new pressure field have been calculated from Eq. (65). The velocities are corrected using Eq. (59). However, only the pressure gradient term is considered, while the corrections from neighbouring velocities is neglected. It is, therefore, necessary to correct the $\mathbf{H}(\mathbf{U})$ term, formulate the new pressure equation and repeat the procedure. Thus, the PISO loop consists of an implicit momentum predictor followed by a series of pressure solutions and explicit velocity corrections. This loop is repeated until a pre-determined limit is reached.

In the `PIMPLE` algorithm, the PISO algorithm is looped within one time-step. The number of PISO loops within a time step is controlled by the `nOuterCorrections`, where 1 corresponds to the PISO algorithm. Between the loops within a time-step, under-relaxation is permitted, in contrast to the regular PISO. `PIMPLE` also allows for an adjustable time-step, which is executed through local time stepping, presented in the following section. Combining the adjustable time-step with looping of the PISO algorithm within a time-step, `PIMPLE` should be a robust algorithm allowing for large time-steps in order to quickly get the flow problem to a steady state.

5.2.5 Local Time Stepping

Time discretization suffers from the limitations of the time step due to stability reasons, as the Courant number restricts the size of the time step through the local cell size. This is especially true for a problem such as ship hydrodynamics, where the cell size varies substantially throughout the domain. When a global time step is used, it has to fulfil the restrictions defined through the smallest cell size. Hence, the global time step becomes very small. The basic idea for local time stepping is to circumvent this problem by using large time steps to advance the solution at large scales, while small time steps are used to advance the solution at fine scales. This will reduce the number of operations if fine scales are only required locally [39]. The reduction of operations will substantially reduce the computational costs, but is only applicable to steady-state problems.

In `OpenFOAM` the solver with local time stepping for a multiphase problem is named `LTSInterFoam`. In `OpenFOAM`, the time step will first be maximized according to the local Courant number. Then the time-field is processed by smoothing the variation in time step across the domain to prevent instability due to large conservation errors caused by sudden changes in time step; spreading the most restrictive time step within the interface region across the entire region to further reduce conservation errors. [34]

5.2.6 Free-Surface Treatment

As has already been shown is Sec. 3.4.1, it turns out that the treatment of the free-surface is not trivial. In order to capture the free-surface accurately, the advection scheme used has to be compressive by its nature and in addition fulfil the boundedness criterion. In `OpenFOAM`, this is achieved through an approach where the continuity equation of the volume function Eq. (30) is rewritten as

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (U\phi) + \nabla \cdot (\phi(1-\phi)U_r) = 0, \tag{67}$$

where U_r is an artificial velocity field which is defined in the vicinity of the interface, in such a way that the local flow steepens the gradient of the volume fraction function and the interface resolution is improved. The last term of the above equation is a compressive term, and it is only active at the interface. This compressive term is discretized using the interface compression scheme `interfaceCompression`, described in [40]. The convective term is treated with the Van Leer second-order Total Variation Diminishing scheme [41], `vanLeer`. Thus, the free-surface is treated with a compressive, stable and second-order accurate method.

5.2.7 Linear Solvers

The previously presented discretization of the governing equations will result in a system of algebraic equations. By following the notation used in [6] a general equation system can be written as

$$A\phi = Q. \tag{68}$$

In CFD, these equation systems are very large and sparse, meaning that there are only a few non-zero elements in the system above. Such systems require special solution algorithms, and the ones used in this work will briefly be presented in this chapter. These are the Preconditioned Biconjugate Gradient (PBiCG) and the geometric-algebraic multi-grid (GAMG). The first one is used for the solution of the velocities as well as the turbulent properties, while the latter is used for the pressure equation. The PBiCG method is based on the conjugate gradient method, which in turn is based on the fact that solving Eq. (68) is the equivalent to the problem of finding the minimum of a new function F

$$F = \frac{1}{2}\phi^T A\phi - \phi^T Q. \tag{69}$$

The function is minimized with respect to multiple directions simultaneously while searching in one direction at a time. In practice, every new direction is perpendicular to all the previous ones and the error is reduced at every iteration. The rate of convergence depends on the condition number τ of the matrix as follows

$$\tau = \frac{\lambda_{max}}{\lambda_{min}}, \tag{70}$$

where λ_{max} and λ_{min} are the largest and smallest eigenvalues of the matrix. For a typical CFD problem, this condition number is very large and the system will thus suffer from slow convergence. One method to increase the rate of convergence is to lower the condition number through preconditioning. Here the initial problem is replaced with another one which has the same solution as the original, but a smaller condition number. This can be achieved by multiplying the original equation with a matrix, so that the resulting equation has the same solution but different eigenvalues.

As such, the conjugate gradient method is only applicable to symmetric problems. As seen in previous chapters, the systems encountered in CFD are in general not symmetric. Thus, the asymmetric systems have to be converted into symmetric ones. This can be done for example by increasing the size of the original system as follows

$$\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \cdot \begin{pmatrix} \psi \\ \phi \end{pmatrix} = \begin{pmatrix} Q \\ 0 \end{pmatrix}. \tag{71}$$

The system now consist of two subsystems, the original system and an additional one which is irrelevant. Thus, the system has increased in size and has become symmetric. The preconditioned conjugate gradient method can now be applied to this system, this is the PBiCG.

The pressure equation is solved with GAMG. The method is based on the fact that iterations are computationally more friendly when performed at coarser grids. So the idea behind a multi grid solver is to use a coarse grid with fast solution times to smoothen out high frequency errors and to generate a starting solution for the finer grid. This can be achieved either by a geometric coarsening of the grid (geometric multi-grid), or regardless of the geometry by applying the principles directly to the matrix (algebraic multi-grid). The strength of GAMG lies in the fact that the iterations are performed much faster at the coarse level. The computational time is (almost) only dependent on the number of unknowns, and not the size of the problem itself.

The original grid is first coarsened step by step and then refined while the equations are solved on each grid level. Mapping onto a finer grid is known as prolognation, while the other way around is known as restriction. Both of these can be done in a number of different ways, with the simplest one being linear interpolation. Before mapping any solution to another grid level, the solution has to be smooth. This is achieved by using an iterative method which produces smooth solutions, such as Gauss-Seidel. A basic two-level method consists of the following parts [6]

1. Perform the iterations on a fine grid
2. Compute the residuals on this grid
3. Restrict these residuals onto the coarse grid
4. Iterate the correction equation on the coarse grid
5. Prolognate the correction to the fine grid

6. Update the solution on the fine grid

7. Repeat the whole loop until desired accuracy is achieved.

In practise, the procedure should continue to a very coarse grid, as the number of unknowns on the coarsest grid is so small that the equations can be solved at an negligible cost. In this work, coarsening was done by one level at the time (`MergeLevels`) to a coarsest level of 1000 cells (`nCoarsestCells`). For smoothing, the default `OpenFOAM` method `GaussSeidel`, was used.

6 Case Setup

In order to run a CFD simulation, a set-up of the problem has to be completed first. The first step is to generate a grid around the original hull geometry. After this, the boundary conditions are imposed, wherafter the computations will commence.

6.1 The Hull

The hull studied in this work is that of a cruise ferry. It has a slender body with a surface piercing bulbous bow. The stern of the hull maintains almost constant breadth and she has a centreline skeg. Model scale towing tank results are available. The geometry has been provided by STX Finland. Table 3 describes the hull geometry, while the concept drawings of the hull are presented in Fig. 14.

Table 3: Dimensions of the hull.

Size	Full scale	Model scale
Scale	1.0	1 : 22.713
Main particulars		
$L = L_{PP}$(m)	200.8	8.8496
L_{WL}(m)	204.8	9.0190
B_{WL}(m)	35.0	1.5410
T(m)	6.8	0.2994
S(m^2)	7815.4	15.1497
Test conditions		
U	21.0 kn	2.2668 m/s
Re	$2.21 \cdot 10^9$	$20 \cdot 10^6$

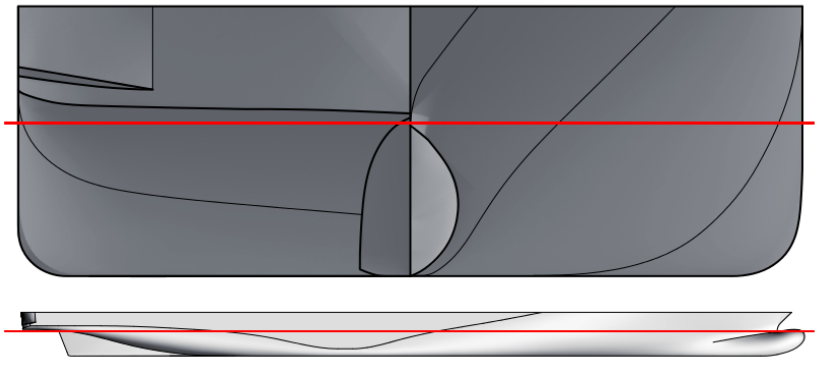

Figure 14: Hull concept drawings, water line indicated in red.

6.2 Grid Generation from the Geometry

The geometry described in the previous chapter was provided in IGES format. After performing a check to make sure that the file is clean, i.e., there are no connectivity failures or other geometrical problems, it was imported to the grid generation software.

6.2.1 Grid Generation in HEXPRESS™

The initial approach of using the built-in grid generation utilities `blockMesh` and `snappyHexMesh` of `OpenFOAM` was abandoned after extensive research, as an satisfactory grid was not obtained. The grid generation was instead done with HEXPRESS™ software, distributed by NUMECA International. For this software, the original IGES file had to be converted to Parasolid format [42].

The grid generation in HEXPRESS™ was done through the following four steps. Firstly, a background mesh was built around the ship hull. Secondly, the grid was snapped around the hull geometry. Thirdly, refinements at the hull and the water-air interface were made in order to accurately capture the flow close to the ship hull and the free surface, respectively. Finally, layers around the hull were added in order to capture the viscous effects. Exact parameters for the steps described above can be found in Appendix A.

6.3 The Grid

Both the double hull and the free surface simulations were done on an identical set of background meshes. The same background grid, with the dimensions of $7 \cdot L_{PP} \times 2 \cdot L_{PP} \times L_{PP}$ was used for the double hull simulations, whereas a grid of $7 \cdot L_{PP} \times 2 \cdot L_{PP} \times 2 \cdot L_{PP}$, was used for each free surface case. The longitudinal direction of the hull was set in the x-direction, while the beam direction was set in the y-direction. The stern of the ship was placed at $x = 0$, while the free surface was set to $z = 0$.

The original aim was to double the amount of cells in the background mesh for each refinement. In practice, this would have been achieved by using a multiplier of $2^{1/3}$ in each spatial direction. However, the simulations for the most finest grids obtained through this refinement method were found not to converge. Thus, the last two refinements of the background mesh were done with a factor smaller than the original $2^{1/3}$. This same approach was extended to the height of the first cell. The amount of cells in z-direction was always set to an even number in order to ensure that the free surface at $z = 0$ would coincide exactly at the line between two cells and leaving an equal amount of cells on both sides of the free surface line. A schematical view of the domain is given in Fig. 15, while the final number of cells as well as the height of the first cell for each grid are presented in Table 4.

In order to capture the free surface accurately, different refinement approaches around the free-surface were employed. The initial approach was to only have refinements in the vicinity of the hull and the wake. However, such refinements in the

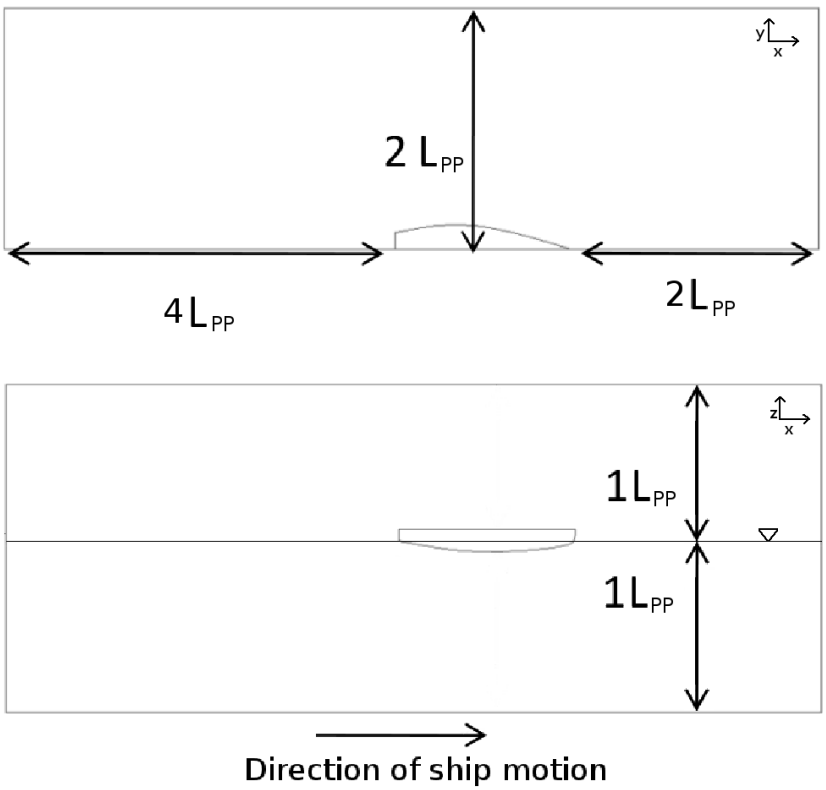

Figure 15: Dimensions of the computational domain.

Table 4: Number of cells and 1st cell height for each case.

Case	Double Hull (10^6)	Free-Surface (10^6)	1st Cell Height (10^{-3}m)
1	0.14	0.62	2.8
2	0.21	1.04	2.2
3	0.35	1.72	1.8
4	0.40	1.89	1.7
5	0.44	2.16	1.6

z-direction caused erratic behaviour of the free surface. Another problem encountered were waves reflected backwards from the outlet. Thus, the final solution was to begin refinements in the z-directions at the inlet, and extended to $-3 \times L_{PP}$. For the wave pattern to be captured, refinements in x- and y-directions were made around the ship hull and wake. These refinements were done between the points $(-2 \times L_{PP}, 0)$ and $(1.5 \times L_{PP}, L_{PP})$. Finally, the domain was extended from $-3 \times L_{PP}$ to $-4 \times L_{PP}$ without any refinements in order for all waves to dissipate and thus not reflect from the outlet. Figure 16 depicts the refinements of the free surface. A comparison between the coarsest and finest grids around the bow is given in Fig. 17.

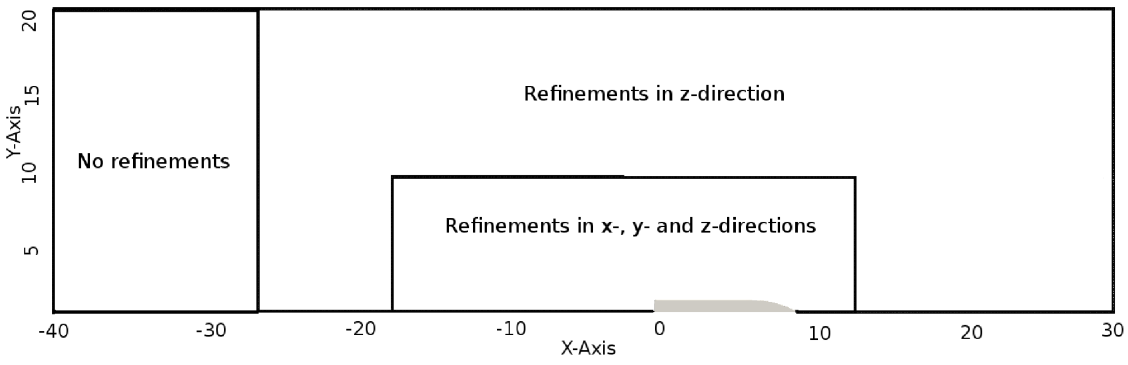

Figure 16: Free surface refinements around the ship hull.

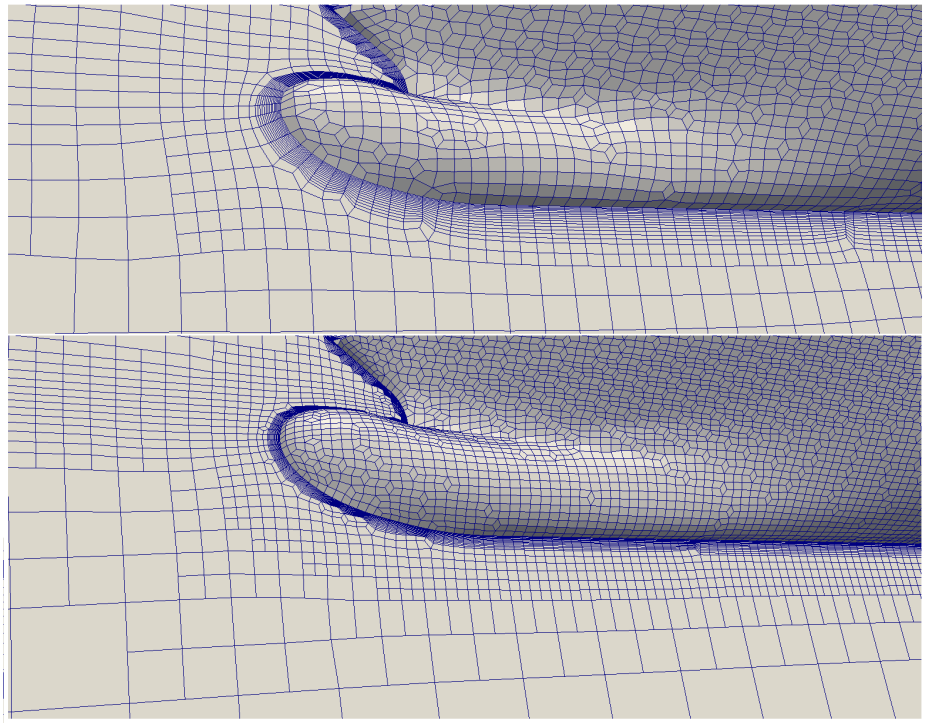

Figure 17: The mesh around the bow for the coarsest grid (top) and the finest one (bottom).

6.4 Boundary conditions

In `OpenFOAM`, the boundary conditions are implemented on the patches of the domain. All simulations were done with the same set of patches and boundary conditions. Figure 18 shows the domain patches for the double hull cases and Table 5 explains the boundary conditions used. The free-surface patches are presented in Fig. 19 and the respective boundary conditions are given in Table 6.

The *fixed values* of Table 5 and 6 are:

- Due to the selected frame of reference, the inlet velocity has the direction $(-1, 0, 0)$. The magnitude is that of the reference velocity, 2.668 $^{\mathrm{m}}/_{\mathrm{s}}$.
- As the pressure only works as a reference pressure, it is set to zero, i.e., $p_\infty = 0$. This is the piezometric pressure.
- The turbulent kinetic energy k and the specific dissipation ω are calculated in Sec. 6.5
- In `OpenFOAM`, α denotes the volume fraction of water in a cell. For the water part of the inlet $\alpha = 1$, while $\alpha = 0$ for the air part of the inlet. In addition, each cell in the internal field is given a value of 1 or 0 during the initialization of the problem.

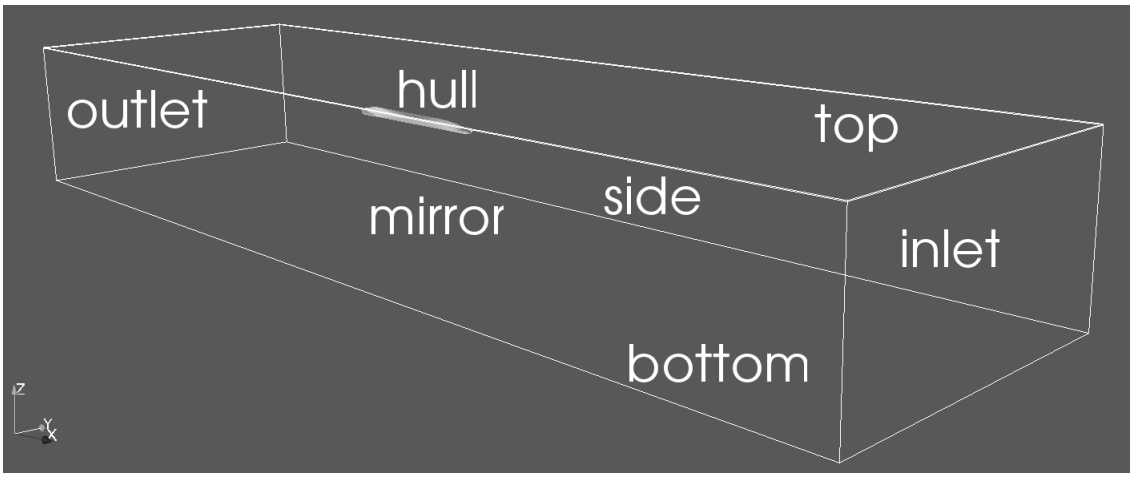

Figure 18: Domain patches for the double hull cases.

Table 5: Boundary conditions used for the double hull cases.

Patch name	U	p	k	ω
inlet	fixed value	zero gradient	fixed value	
top	fixed value	$p = p_\infty$	zero gradient	
bottom	fixed value	$p = p_\infty$	zero gradient	
side	fixed value	zero gradient	zero gradient	
mirror	symmetry			
outlet	zero gradient			
hull	$U = 0$	zero gradient	wall function	
internal field	fixed value			

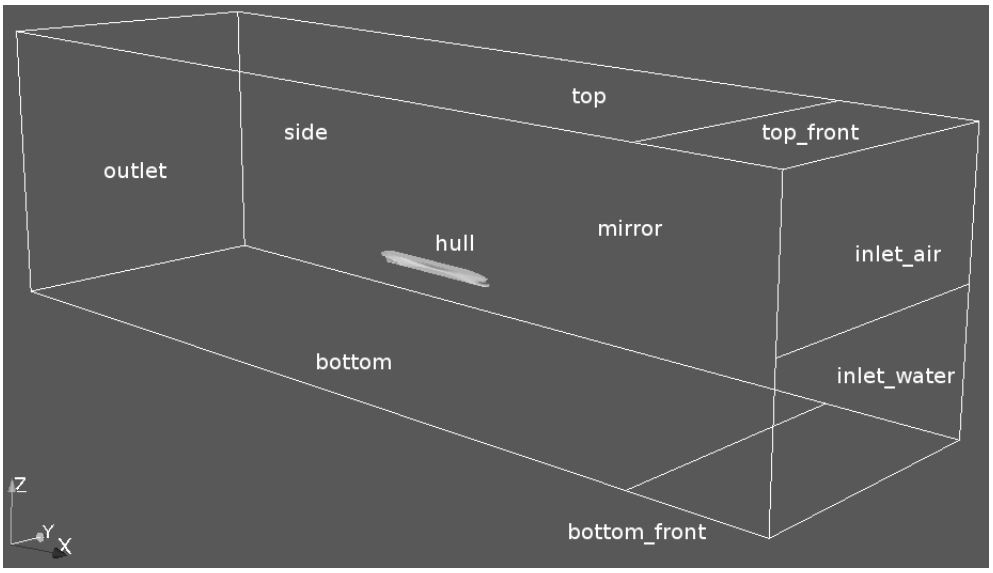

Figure 19: Domain patches for the free-surface cases.

Table 6: Boundary conditions used for the free-surface cases.

Patch name	U	p	k	ω	α
inlet_water	fixed value	zero gradient	fixed value		
inlet_air	fixed value	zero gradient	fixed value		
top_front	fixed value	$p = p_\infty$	zero gradient		
top	fixed value	zero gradient	zero gradient		
bottom_front	fixed value	$p = p_\infty$	zero gradient		
bottom	fixed value	$p = p_\infty$	zero gradient		
side	fixed value	zero gradient	zero gradient		
mirror	symmetry				
outlet	zero gradient				
hull	$U = 0$	zero gradient	wall function		zero gradient
internal field	fixed value				

6.5 Turbulence Properties

The boundary values for k and ω are dependant on the problem geometry. Based on earlier work done at the ship hydrodynamics group [43], the turbulence intensity was set to $I = 3.5\%$. Based on this intensity, the turbulent kinetic energy was calculated as

$$k = \frac{3}{2}(IU)^2 = 0.009442 \ {}^{\mathrm{m}^2}/_{\mathrm{s}^2}. \tag{72}$$

The specific dissipation rate was calculated as suggested by Eça and Hoekstra [44] through the following formula

$$\omega = 10\frac{U}{L_{PP}} = 2.564 \ \mathrm{s}^{-1}. \tag{73}$$

7 Results

In this chapter, the results of this work are presented. First, the results for the double hull case are presented, followed by the results for the free-surface case. For each case a grid refinement study based on the drag coefficient is presented, followed by a presentation of the y^+ and y^+ values. Due to the sparse data available from the measurements, only the total resistance from the free-surface case [46] can actually be compared to the measured results.

In the last section a comparison between the two cases is done in an effort to answer the original research problem. The differences between the flow regimes are first presented qualitatively, through visualizations of the wetted surfaces and streamlines. This is followed by an analysis of the pressure and shear stress distributions, concluding in an quantitative analysis of the shear stresses and pressure coefficients along different cuts of the hull.

7.1 Results for the Double Hull Case

The first computations were conducted for the double hull case. As these calculations were the first ones made, they would also serve as an evaluation tool for the boundary conditions as well as the discretization schemes and solver settings. However, the results for these test cases are omitted in order to make the presentation clearer. The double hull computations turned out to be a relatively straightforward, with only little iterations between settings and meshes. The behaviour of the drag coefficient was found to be stable after 25 s. The convergence histories for the drag coefficient in each case together with other variables from case 5 are presented in Fig. 20.

The grid convergence was studied using five different grid densities. In order to minimize the errors due to spatial discretization, the results are estimated on an infinitely fine grid (zero grid spacing) by extrapolation of results from different grid levels. This was done through Richardson extrapolation, which is the recommended method of the the ITTC [45]. The results from different grid levels should display asymptotic behaviour and the difference between two adjacent grid levels should always decreases when going towards the finer grid. The grid spacing parameter h is normalized relative to the grid spacing on the finest grid. As the amount of cells in the background mesh doubles for each refinement step, the discrete points fall at $h = \{1, 2, 4, 8, 16\}$. However, the two finest grids do not follow this refinement pattern as the computations on these would not converge. Thus, the two finest grids in Fig. 21 are not at $h = \{1, 2\}$.

The drag coefficient for the five different grids as well as the result from the Richardson extrapolation are compared in Fig. 21 with the calculated value from ITTC-57, Eq. (41). The results display asymptotic behaviour, with the Richardson extrapolation overestimating the C_d by over 10 % when compared to the ITTC-57 friction line. However, as this friction line is a function of Re only and does not take into account the hull form at all, the results attained are not necessarily inaccurate.

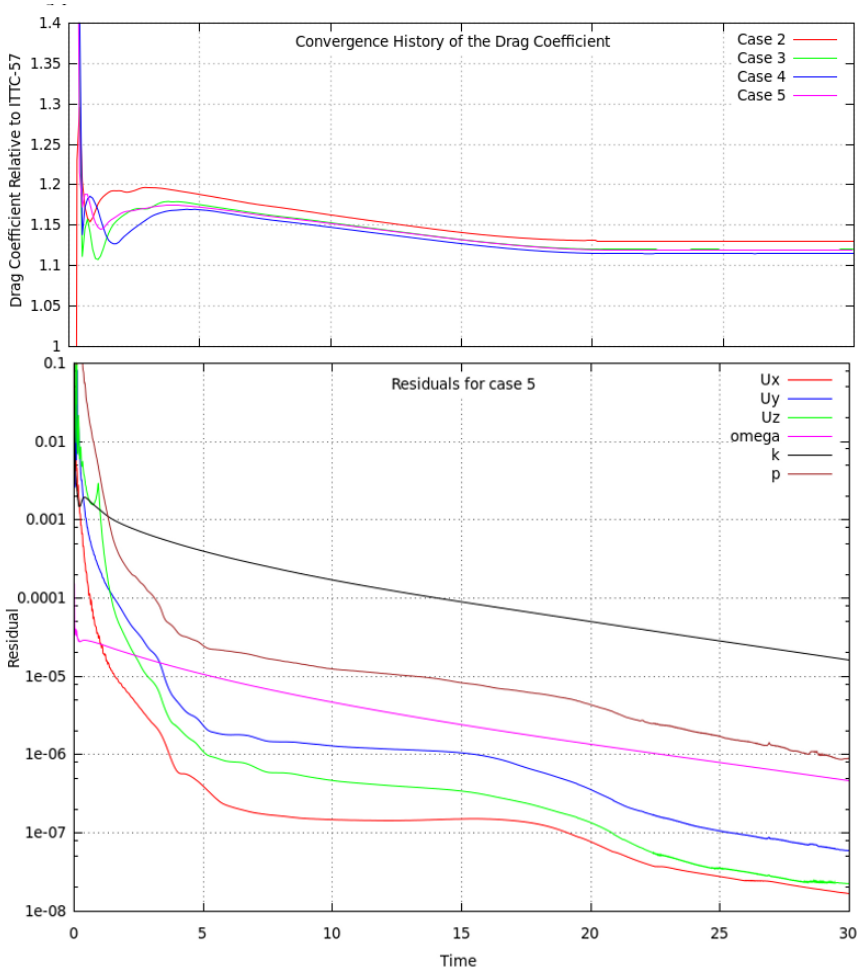

Figure 20: Convergence of the drag coefficient C_d (upper), and case 5 (lower).

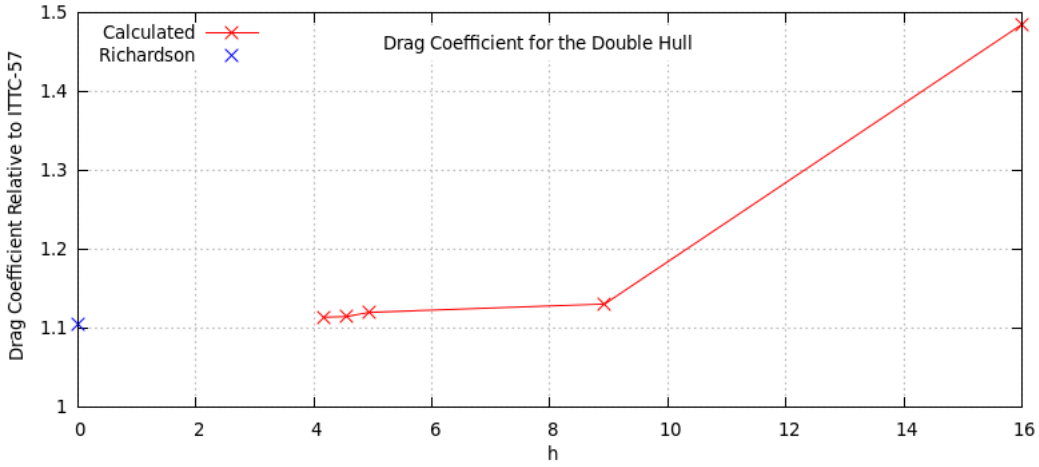

Figure 21: Drag coefficient for the double hull case.

The y^+ values of the first cell next to the wall are important in order for the viscous effects to be captured accurately. As wall functions are used, y^+ values between $30 - 300$ are allowed. The distribution of y^+ is presented in Fig. 22. The values for y^+ are clearly within range, with small exceptions at the bow and stern areas, where the value locally drops to around 10. The boundary layer is further examined in Fig. 23 where the velocity distribution from $x/L_{PP} = 0.5, z//L_{PP} = -0.03$ is compared to that of Eq. 29. The velocity profile in the log-law layer appears to be correct, albeit a 5 % difference in the u^+ values. However, there is a clear error with the velocity at the cell nearest to the wall. A measure of turbulence is presented in Fig. 24, where the ratio between the turbulent viscosity ratio, i.e., ν_t/ν is presented.

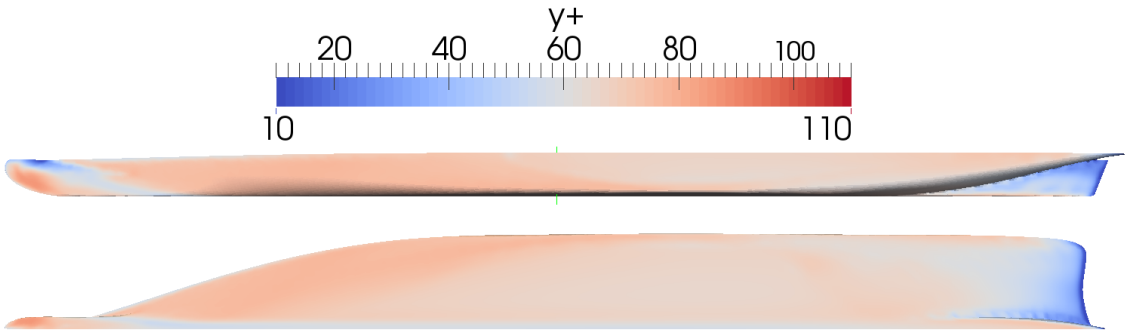

Figure 22: The y^+ distribution for the double hull case.

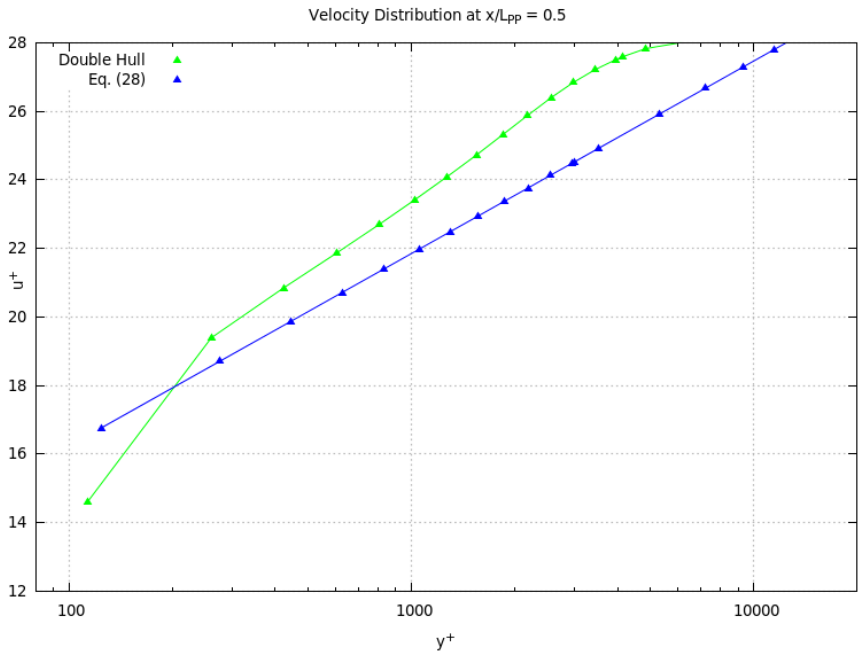

Figure 23: The u^+ distribution in the boundary layer.

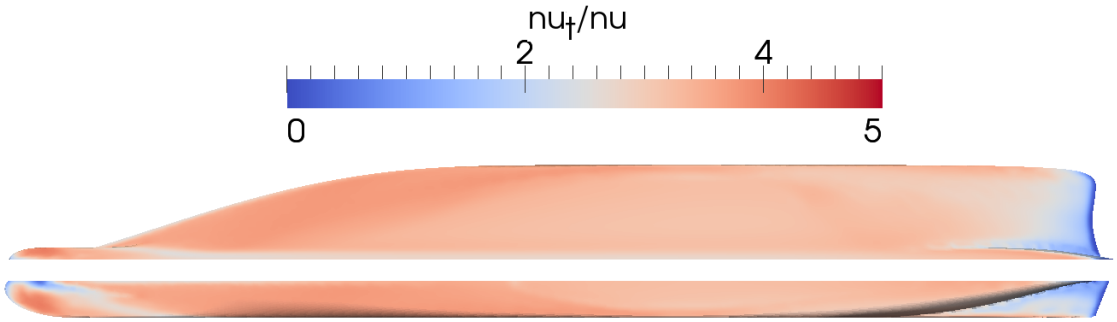

Figure 24: Turbulent viscosity ratio ν_t/ν on the surface.

7.2 Results for the Free-Surface Case

Although the free-surface case is governed by substantially different physics, the findings from the double hull case were used as a starting point for the free-surface computations. The free-surface case turned out, not surprisingly, to be significantly more complicated to calculate. A substantial amount of iterations between different settings and meshes had to be executed before a stable solution was found. Especially the modelling of the free-surface turned out to be cumbersome, and the final methods used for this section might not be ideal but at least somewhat robust. Another problem encountered was the amount of non-orthogonal cells which apparently cause stability issues in `OpenFOAM`. To circumvent these problems, a gradient limiter `cellMDLimiter` was applied. However, even the final set-up used for the computations was extremely fragile and would not converge with slightly modified settings. Figure 25 depicts the convergence histories of the drag coefficient C_d for the cases together with a more detailed convergence history for case 5.

In contrast to the relatively stable double hull cases, the free-surface cases exhibited strong oscillatory behaviour. This oscillatory behaviour was of course expected, as the free-surface experiences a totally non-physical acceleration immediately at the start of the simulation. In order to attain a stable solution from the initially transient problem, the amount of iterations was increased to 45 000. However, the oscillatory behaviour could not be totally eliminated. Thus, an average of the last 5 000 rounds was used to calculate the drag coefficient, while the peak values from the corresponding interval were used as the maximum and minimum values for the error estimate. Special treatment was given to the coarsest grid, which does not even fit in Fig. 25, where the simulation was extended to 60 000 iterations but the solution was still heavily oscillatory.

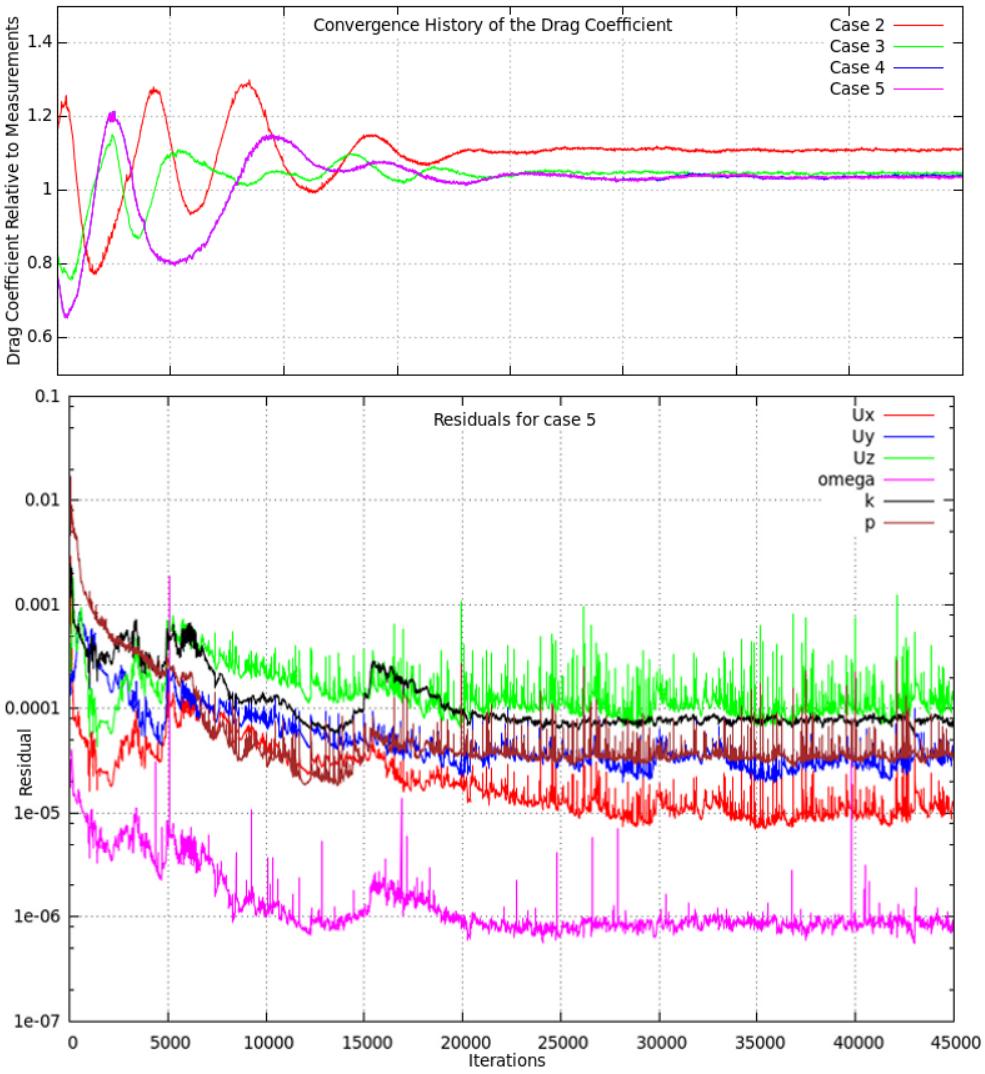

Figure 25: Convergence of the drag coefficient C_d (upper), and case 5 (lower).

A similar grid convergence study as made earlier for the double hull case, was also made for the free-surface case. These results, compared with the measured value of the corresponding hull are presented in Fig. 26. The coarsest grid did not yield a result of any significance, as the oscillations could not be eliminated and thus the error is around $\pm$ 10 % and does not even fit in the figure. Outside of the coarsest grid, the results do display an asymptotic behaviour, with the final result from the Richardson extrapolation overestimating the measured value of C_d by 3.7 %. This can be considered as an acceptable level of accuracy.

Again, the y^+ values are important in order for the viscous effects to be captured accurately. As the same wall functions are used as previously, y^+ values should stay within $30-300$. The distribution of y^+ is presented in Fig. 27. The values for y^+ are

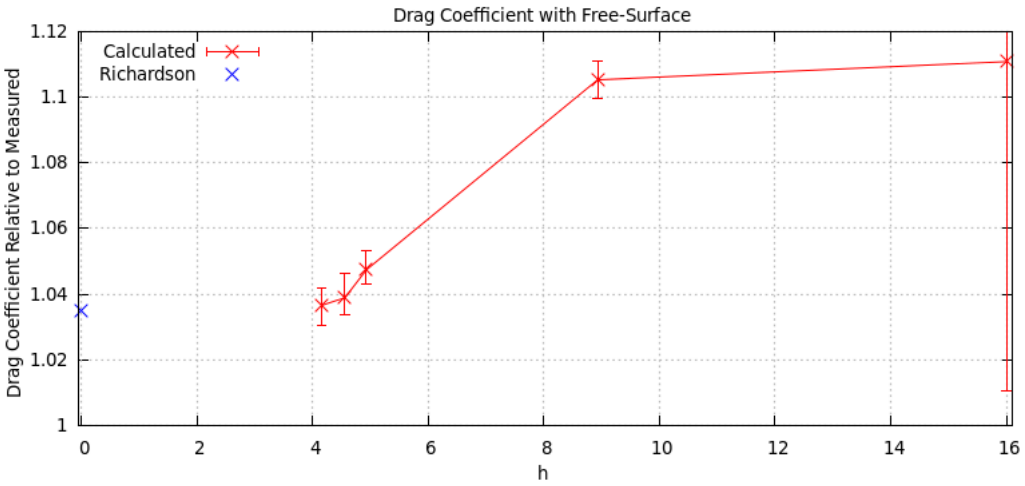

Figure 26: Drag coefficient for the free-surface case.

clearly within the range, with small exceptions at the stern areas, where the value locally drops to around 10. As the heights of the first cells are the same as in the double hull case, y^+ values at the bulb rise higher because of the large shear stresses caused by the surface piercing bulb. The boundary layer is, again, further examined in Fig. 28 where the velocity distribution from $x/L_{PP} = 0.5, z//L_{PP} = -0.03$ is compared to that of Eq. 29. The distribution follows that of Eq. 29, but there is a 5 % difference in the u^+ values. Again, the velocity at the first cell is clearly wrong.

The somewhat erratic behaviour of y^+ in the bottom and near the stern area is explained by the fact that some air has been trapped at the surface of the hull. This is illustrated in Fig. 29 by values smaller than unity on the hull. The trapped air is believed to have its roots in the initial transient phase of the simulation, when nonphysical accelerations govern the domain. A measure of turbulence is presented in Fig. 30, where the ratio between the turbulent viscosity ratio, i.e., ν_t/ν is presented.

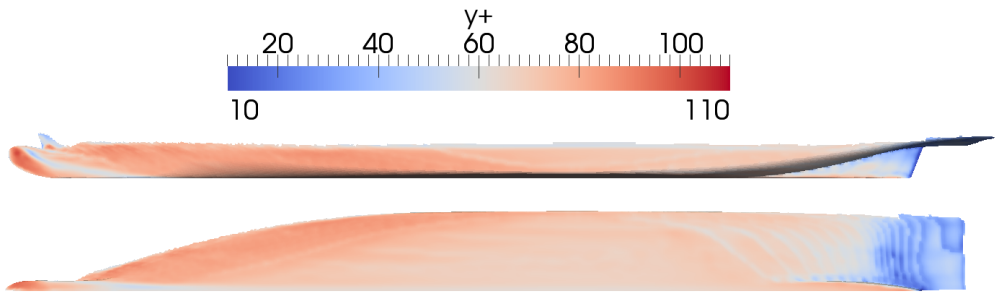

Figure 27: The y^+ distribution for the free-surface case. Only the wetted surface is considered.

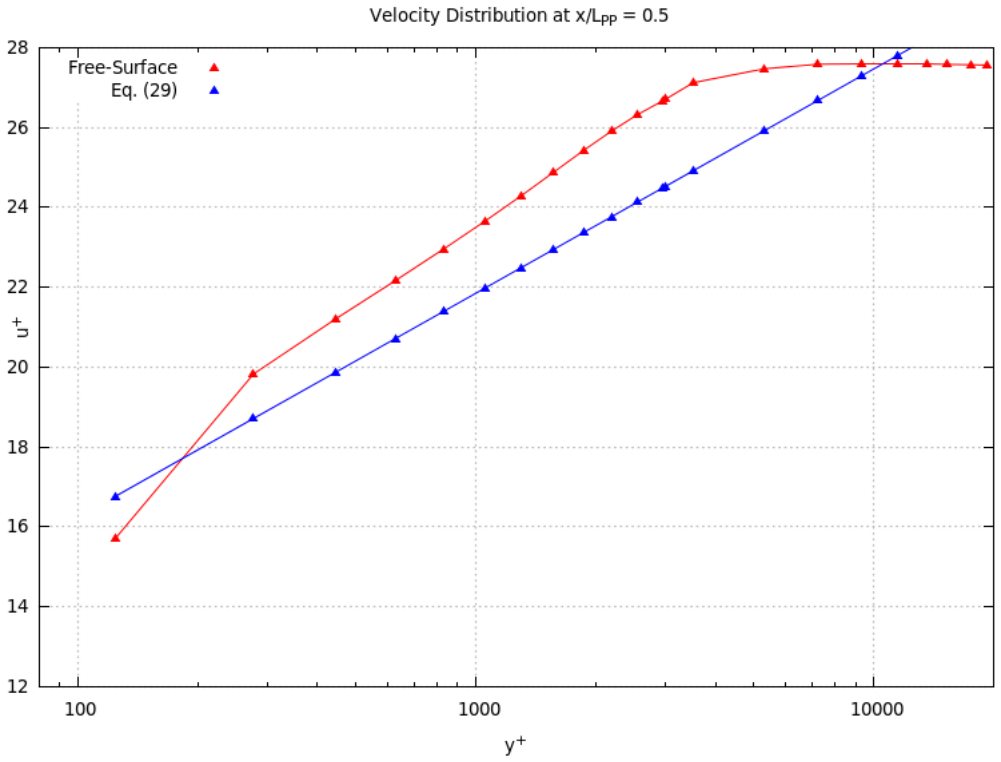

Figure 28: The u^+ distribution in the boundary layer.

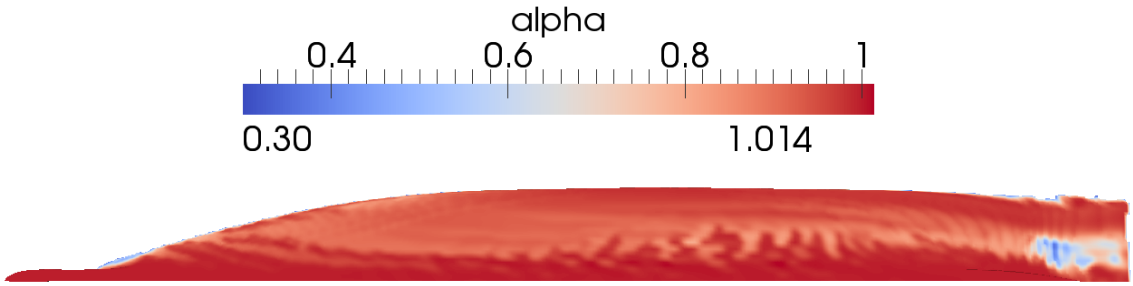

Figure 29: The distribution of volume fraction α on the hull.

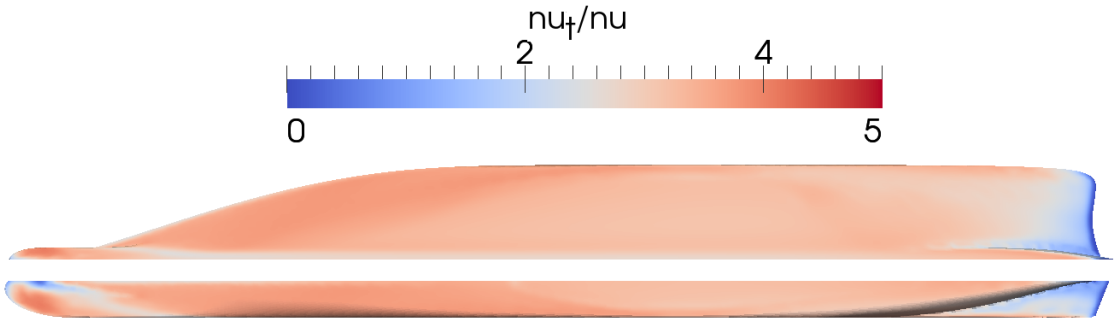

Figure 30: Turbulent viscosity ratio ν_t/ν on the surface.

The wave profile generated by the hull is only considered qualitatively here, as there is no data available from test cases. The wave pattern is presented in Fig. 31. The pattern clearly follows that of the Kelvin wave pattern, presented earlier in Fig. 10. The wave profile at the symmetry line ($y = 0$) is depicted in Fig. 32. Qualitatively, it appears that the computations have been able to capture the wave profile correctly. As explained in Sec. 4.5.2, large waves are created both by the bow and stern, while the fore and aft shoulders create clear troughs. The fluctuations on the hull are a result of uneven meshing in the z-direction. This can be seen in Fig. 17 as the cells on the hull are not aligned in the z-direction the VOF modelcannot capture the interface between water and air accurately.

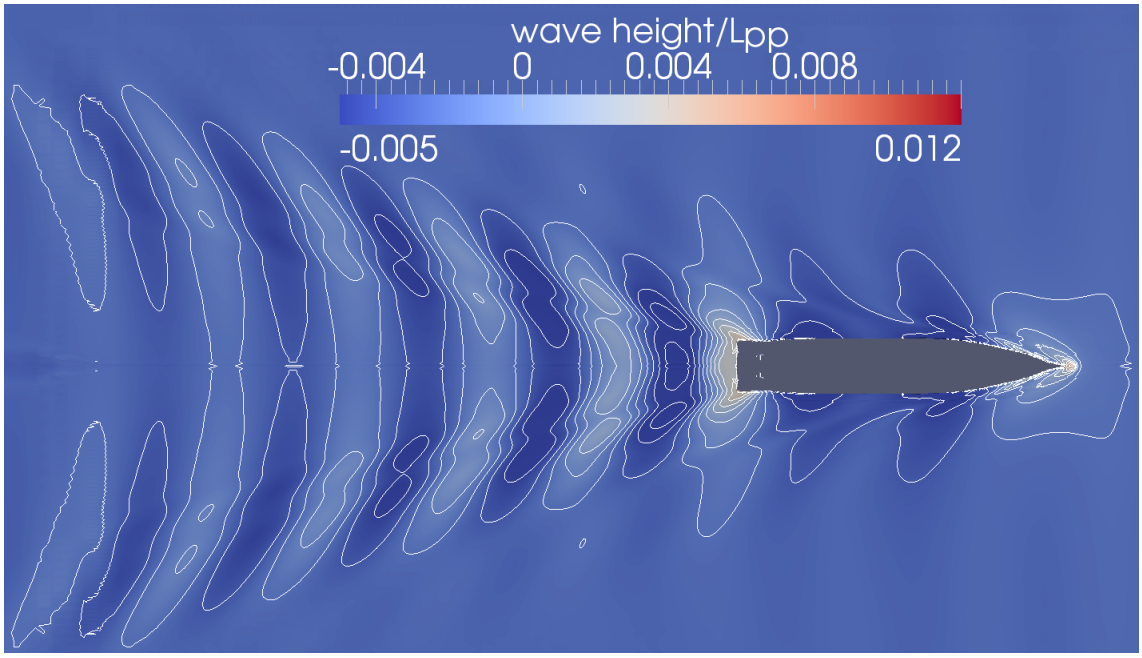

Figure 31: The wave profile.

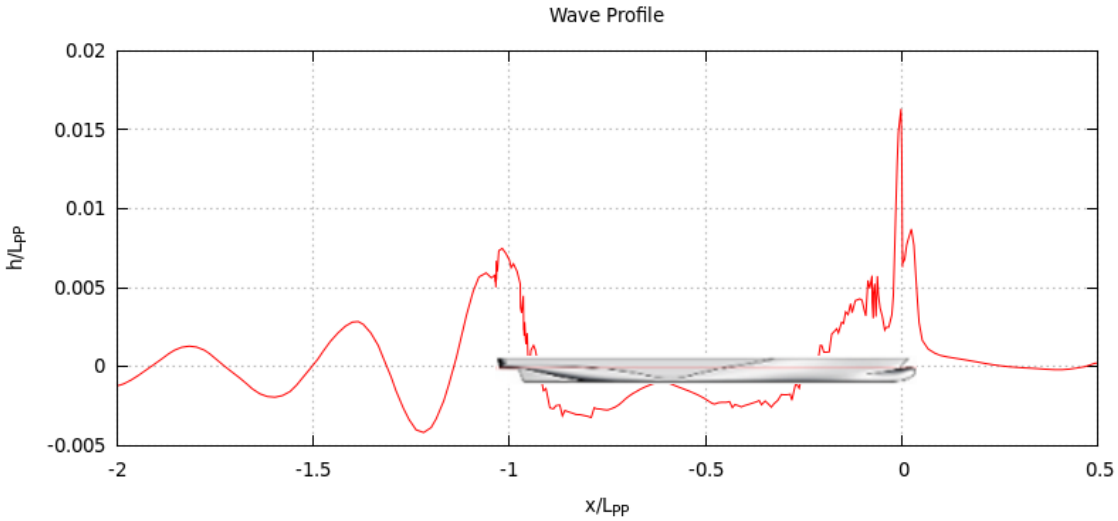

Figure 32: The wave profile on the hull and symmetry line.

7.3 Comparison Between the Cases

In this section, the flow characteristics of the two cases are compared. The effect the free-surface has on the flow regime is clear, and the comparison between the cases will thus begin by investigating the wetted surfaces. A comparison between the wetted surfaces of the double hull and free-surface cases is given in Fig. 33. Not surprisingly, there is a clear difference between the two. While the whole water line is different from the simple straight line for the double hull, the most dramatic changes caused by the free-surface are found in bow and stern waves. The midship is characterized by the fore and aft shoulder systems. Finally, the free-surface raises along the bottom of the stern, resulting in a completely different flow regime in this area. The actual wetted area for the free-surface case is 7.835 m^2, in contrast the nominal area of 7.575 m^2.

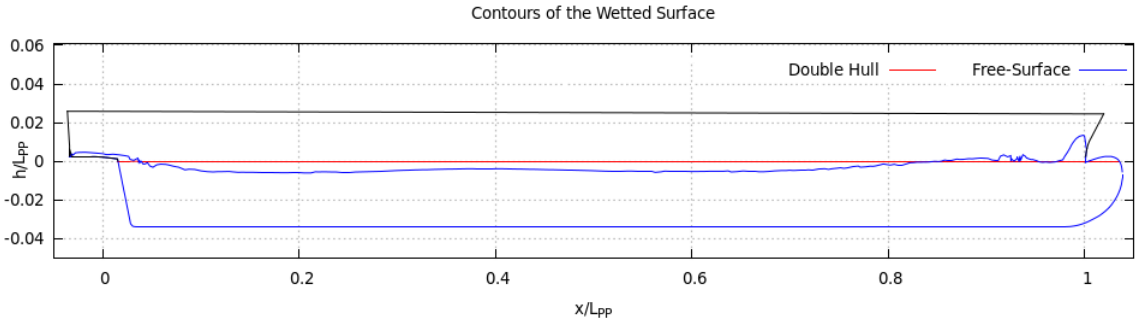

Figure 33: Comparison between the wetted surfaces.

The free-surface effects on the flow regime can be qualitatively presented through the use of streamlines. This is done in Fig. 34, where streamlines for the two cases are compared. Clearly, the flow around the bulbous bow is substantially effected by the free-surface. Whereas the flow around the double hull simply passes the bulb, the situation is more complicated for the free-surface case. The fluid is forced to flow over the bulb, giving rise to a wave system. This wave system influences the whole flow regime, also forcing streamlines away from the hull, as can be seen in the lower part of Fig. 34. Streamlines diverge from the fore part much more than in the double hull case. The corresponding projections for the stern are presented in Fig. 35. Once again the free-surface clearly complicates the flow regime with the wave system being generated. The divergence away from the hull is however not as clear as it is for the bow part.

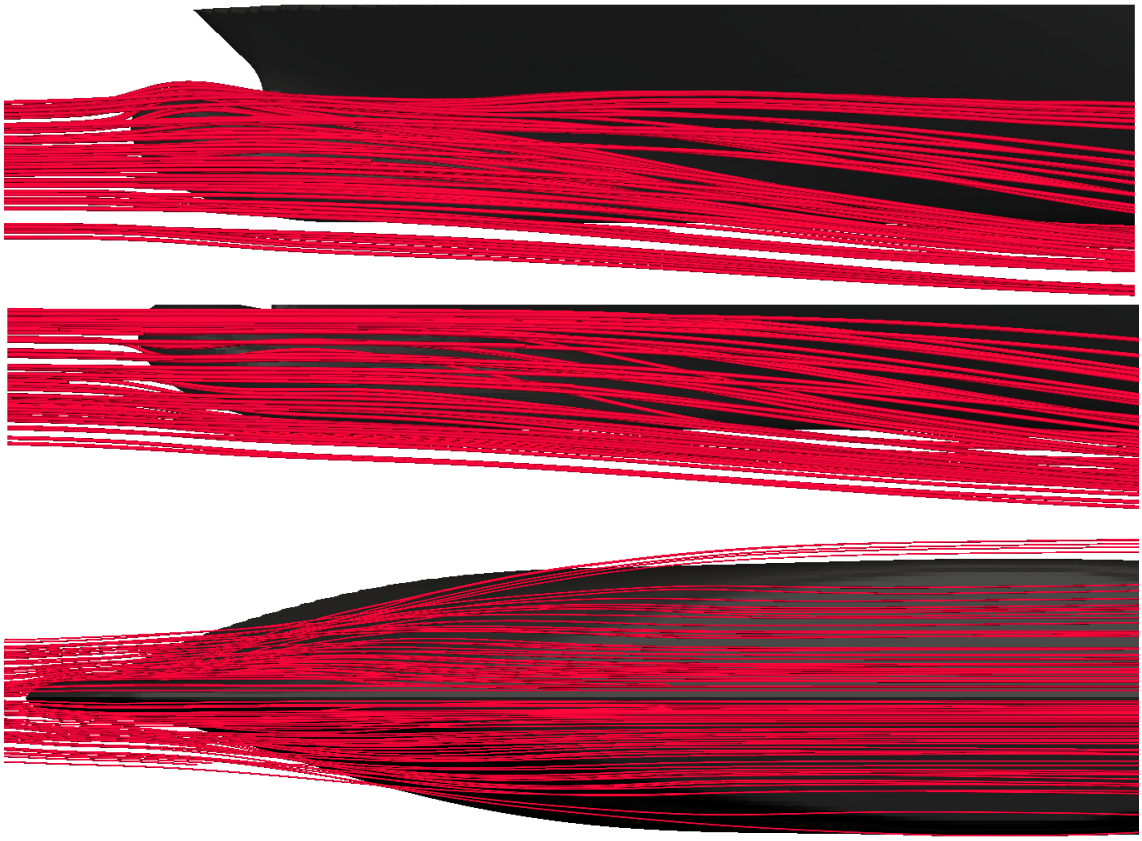

Figure 34: Streamlines from the bulb for the free-surface case (upper) and the double hull case (lower).

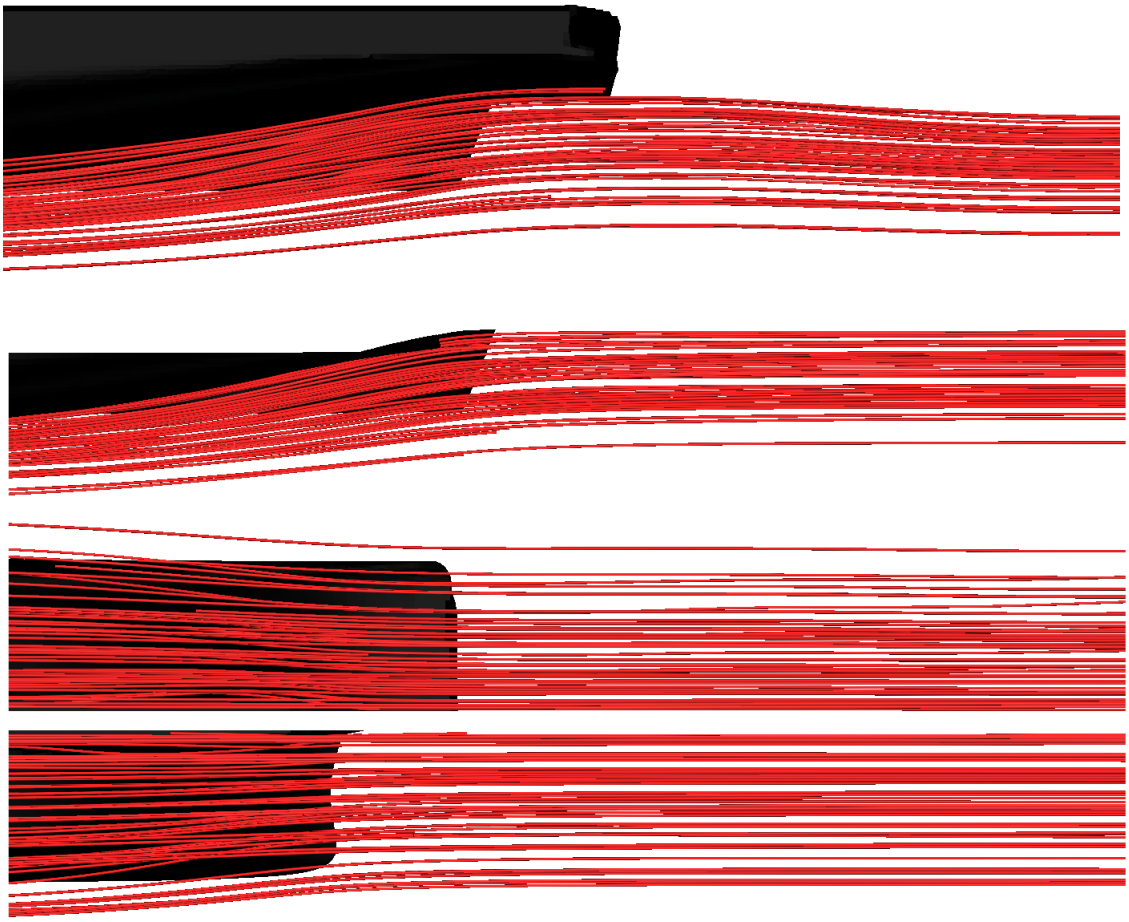

Figure 35: Streamlines in the stern for the free-surface case (upper) and the double hull case (lower).

7.3.1 Shear Stresses

The distribution of shear stresses is presented in Fig. 36. The free-surface clearly influences the shear stresses close to the waterline. This is due to the wave making, which causes larger velocity gradients at the hull. The friction coefficient peaks at the side of the bulbous bow. In this region the shear stresses are once again greater for the free-surface case. The differences between the two cases are not as obvious in the stern region. For the comparable parts, i.e., the region where the free-surface has not risen along the stern, it appears that the shear stresses for both cases behave homogeneously.

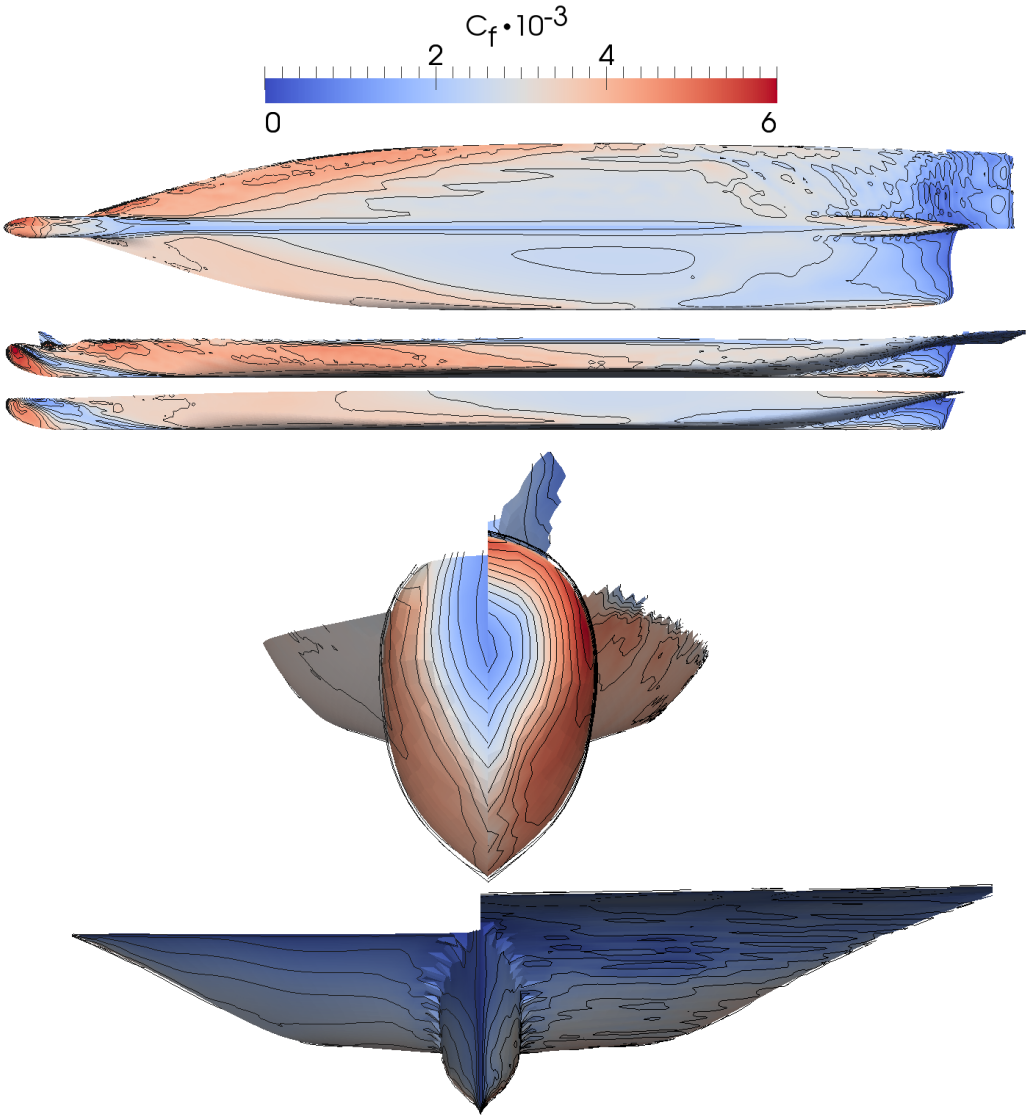

Figure 36: Comparison of the skin friction coefficients at different projections, free-surface case (upper) and the double hull case (lower).

In Fig. 37 the evolution of the skin friction coefficient is plotted along three different waterline cuts, and the results are compared to the formula of Kestin and Persen, Eq. 34. The findings made earlier are supported by Fig. 37. In fact, a number of observations can me made. Firstly, the largest differences between the cases can be found in the bow region of the hull where local discrepancies of around 70 % occur. Secondly, the distribution along a line displays similar shapes for both cases, but the shear stresses for the free-surface case are substantially larger. An exception to this being the $z/T = -0.2$ line, where the free-surface case displays smaller shear stresses for $x \leq 0.7 \cdot L_{PP}$. Finally, all the lines clearly overestimate Eq. 34 at the midship where the flow should be reminiscent to that of a flat plate. Interestingly, by studying line $z/T = -0.2$ for the free-surface case, a narrow region going through the bottom of the hull with clearly lower shear stresses can be found. This region is also visible in Fig. 39. It appears as if a corridor of low velocities would pass the bow region and continue along hte bottom of the hull all the way to the stern region.

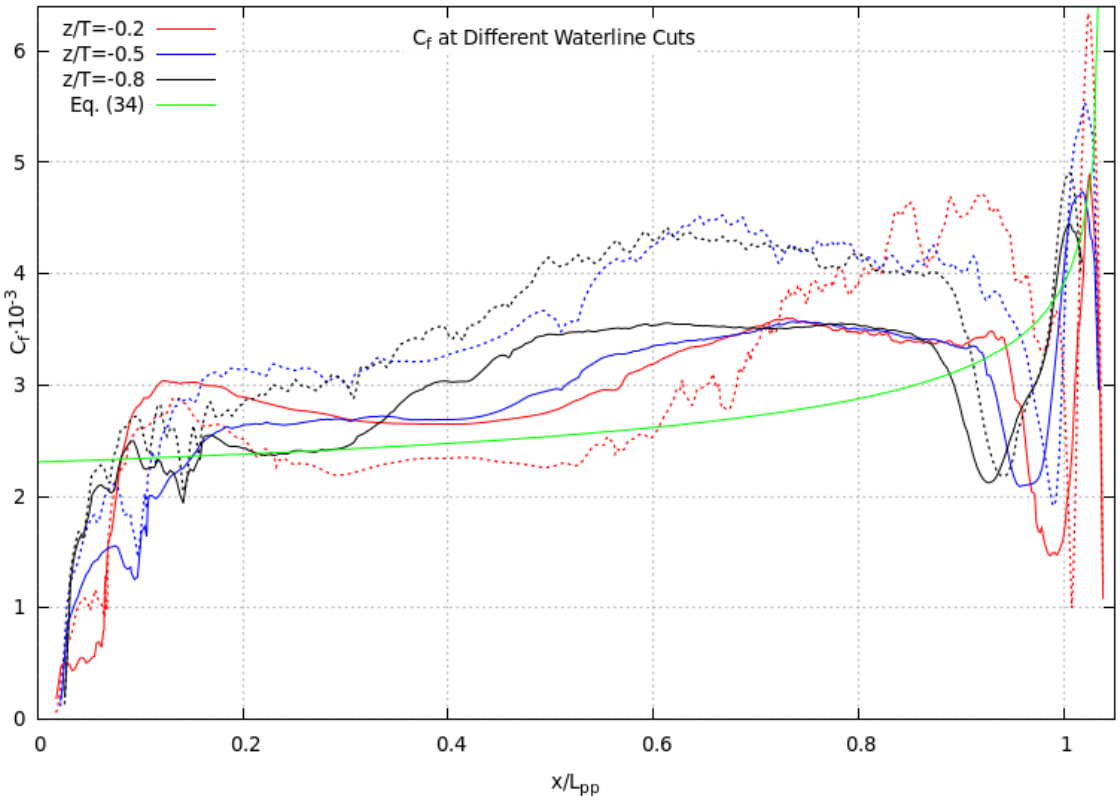

Figure 37: Shear stresses along three different waterline cuts, solid lines represent the double hull case, while the free-surface case is denoted with dotted lines.

The evolution of the skin friction coefficient among five different longitudinal cuts together with Eq. 34 is presented in Fig. 38. The findings made earlier are supported by the analysis of the longitudinal cuts. Again, the biggest discrepancies are found close to the bow region. The distribution along a line displays similar shapes for both cases, but the shear stresses for the free-surface case are substantially larger. As one would expect, this difference becomes smaller at larger depths. The formula by Kestin and Persen, Eq. 34, gives a more accurate estimation at the midship than for the waterline cuts.

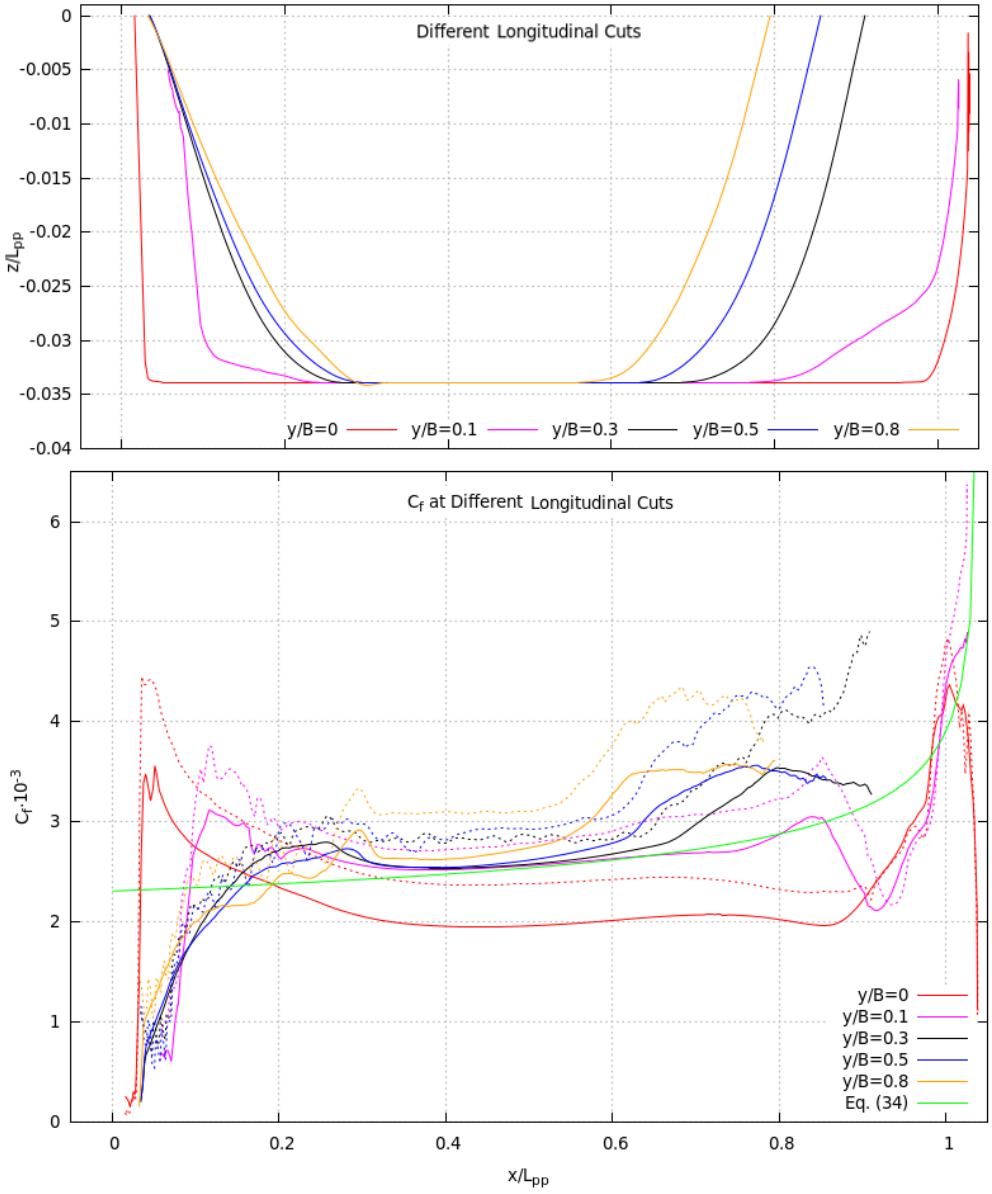

Figure 38: Shear stresses along five different longitudinal cuts, solid lines represent the double hull case, while the free-surface case is denoted with dotted lines.

7.3.2 Pressure Coefficients

The pressure distribution for both cases is depicted in Fig. 39. In both cases the pressure reaches its maximum at the bulb, is evenly distributed at the midship, and grows toward the stern. At the bulb the pressure seems to reach greater values for the double hull case. The pressure throughout the midship is lower for the free-surface case. The pressures seem to behave somewhat identical in the stern region, however the wetted surfaces are different so the results are not directly comparable. The trapped air, explained in Fig. 29, is probably the cause of the vortex system created at the bottom of the hull.

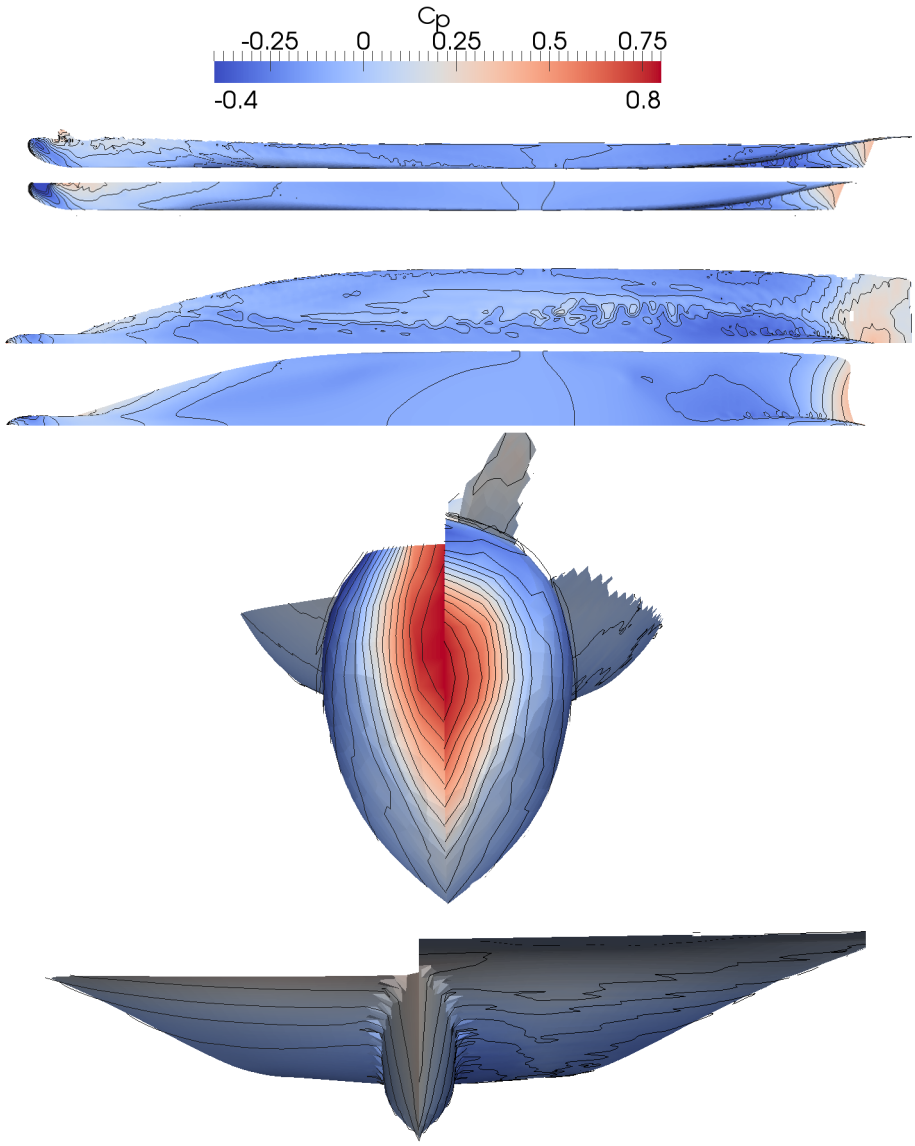

Figure 39: Comparison of the pressure coefficients at different projections, free-surface case (upper) and the double hull case (lower).

In Fig. 40, the evolution of the pressure coefficient is plotted along three different waterline cuts. The qualitative findings made earlier can be supported by a number of findings made here. The pressure coefficients along a line display similar behaviour for both cases. The pressure coefficient reaches its maximum at the bulb, with the double hull producing a maximum value of around 10 % greater than the free-surface. The shoulders clearly cause another peak and once again the double hull produces larger values. At the midship, the pressure is nearly constant within a case, but the double hull displays substantially larger values ($C_p \approx -0.06$) than the free-surface ($C_p \approx -0.13$). Towards the stern the pressure coefficient grows evenly for both cases with the difference between the two staying relatively the same.

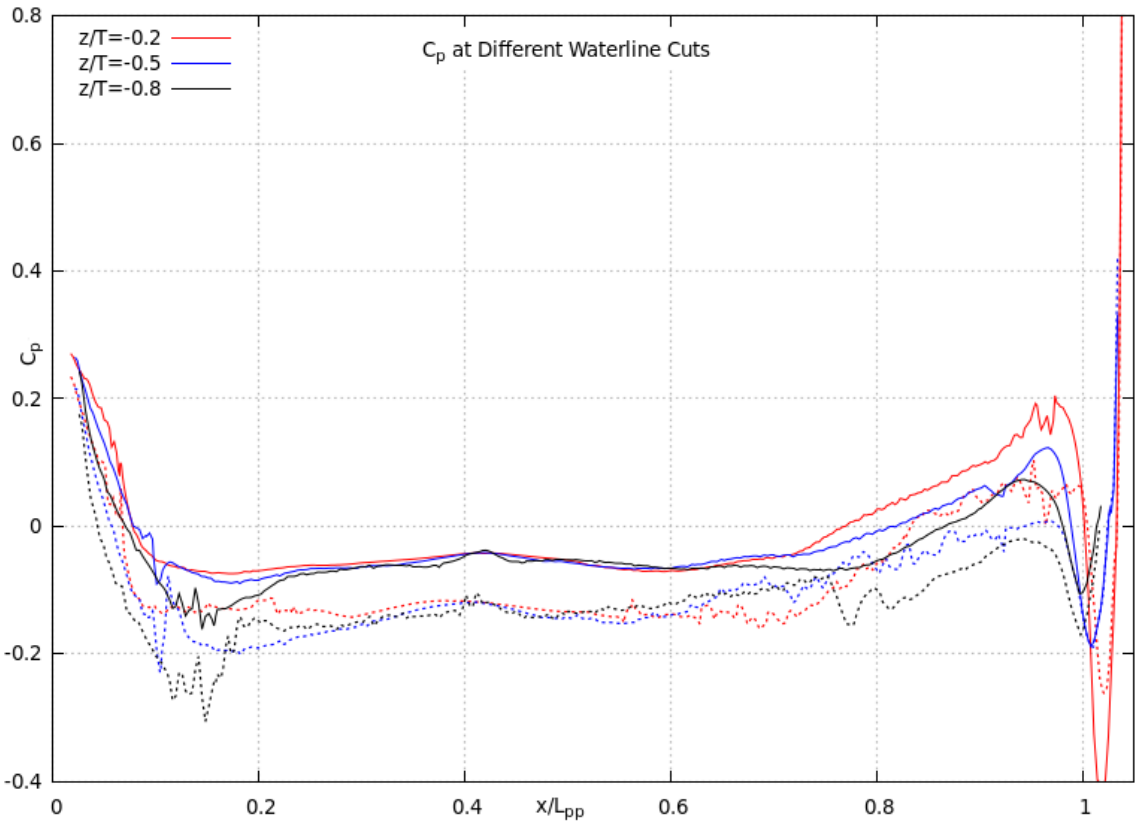

Figure 40: The pressure distribution along three different waterline cuts, solid lines represent the double hull case, while the free-surface case is denoted with dotted lines.

The evolution of the pressure coefficient among five different longitudinal cuts is presented in Fig. 41. The findings made earlier are supported by the analysis of the longitudinal cuts. Once again the pressure coefficients along a line display similar behaviour for both cases, but the double hull gives greater values. However, this difference diminishes with an increase in the y-coordinate, i.e., the further away from the symmetry line one goes. The behaviour of the pressure coefficient at the bulb is nearly identical for both cases, while it at the stern displays similar characteristics

as in Fig. 40.

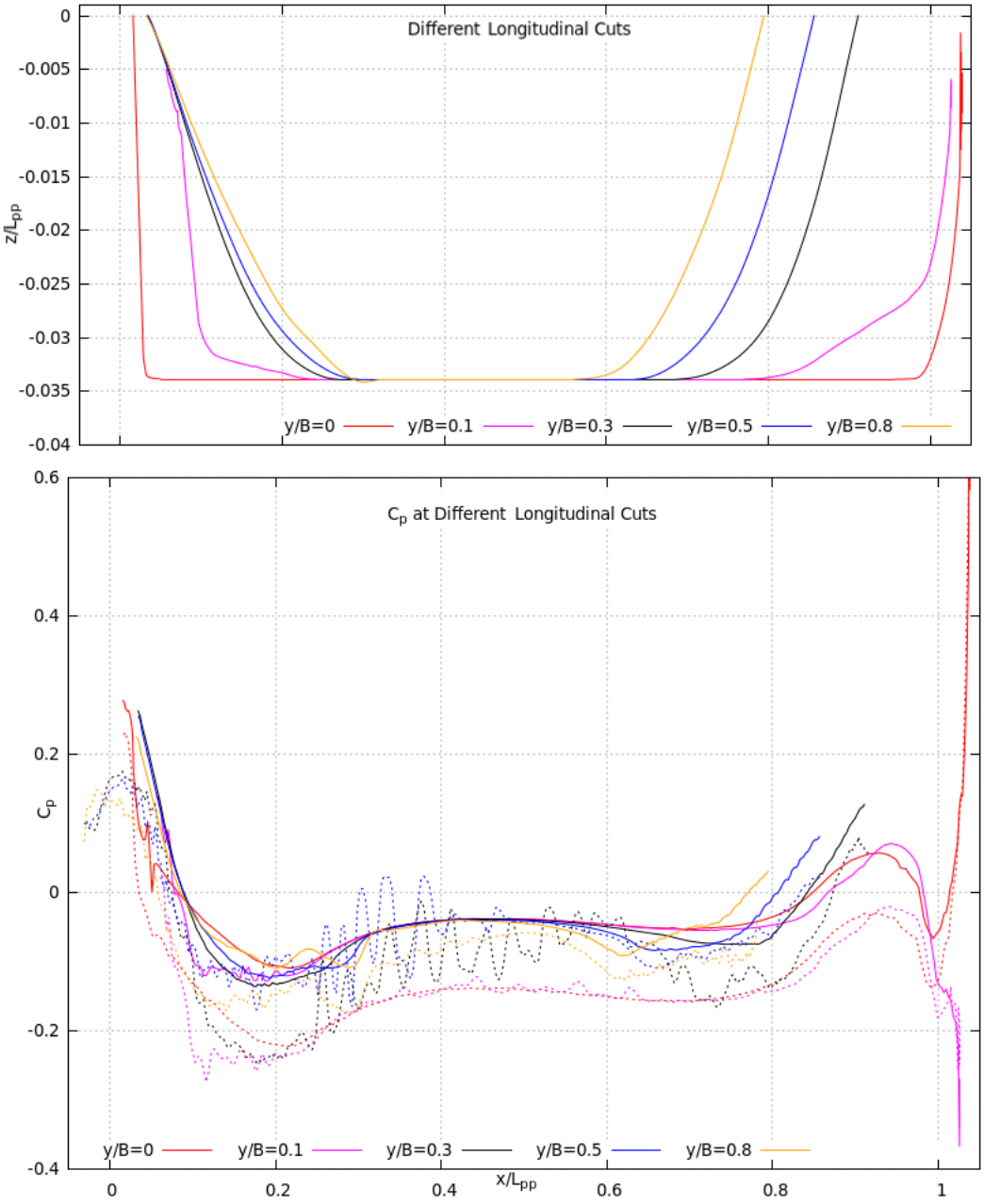

Figure 41: The pressure distribution along five longitudinal cuts, solid lines represent the double hull case, while the free-surface case is denoted with dotted lines.

Finally, the friction and pressure resistance coefficients for both cases are compared. Not surprisingly, the majority of the additional drag caused by the free-surface comes as added pressure resistance. The pressure resistance coefficient doubles after the introduction of the free-surface. As the wetted surface in the free-surface case has a different area, the viscous coefficient is nondimensionalized by

both the nominal and actual wetted surfaces. The friction resistance on the other hand increases by approximately 7 % for the nominal wetted surface and 3 % for the actual wetted surface. As the actual wetted surface is larger than the nominal one, the friction resistance coefficient for the latter is slightly larger.

8 Conclusions and Discussion

A CFD analysis of the flow around a cruise ferry hull has been conducted with `OpenFOAM` software. Computations for both a double hull case as well as a free-surface case were made. The results from each case were compared with each other as well as with theoretical and experimental results. For each case, a grid refinement study consisting of five different densities was done. However, as the highest mesh densities would not converge, this refinement study could not be completed with the planned rate of doubling the background mesh at each step. The finest grids thus contained less cells than originally planned.

A lack of experimental data made it difficult to validate the findings made in this work. Only the drag coefficient for the free-surface case was available from towing tank tests. For this part, the results attained differ 3.7 % from the measured value. This can be consider as an acceptable level of accuracy for the drag coefficient. The drag coefficient for the double hull case exceeds the ITTC-57 friction line by 10 %. However, as this friction line is a function of Reynolds number only, no definitive conclusions can be made from the accuracy of the double hull case.

Other quantities from both cases could only be validated qualitatively. The shear stresses at midship for both cases seem to exceed the values for a flat plate. The pressure distributions are in line with the theory presented in this work. Finally, the wave structure created in the free-surface case appears to be reasonable when compared with the theory part.

To answer the original research question concerning the differences between a double hull case and a free-surface case, results from both cases were compared. The free-surface causes a significantly different flow regime around the ship hull, when compared to the double hull case. This is evident when studying the evolution of shear stresses along different cuts on the hull. The shear stresses are clearly larger for the free-surface case. This is especially true the close to the bow area. Also the global friction resistant coefficient is notably larger for the free-surface case. Somewhat surprisingly, the shear stresses at the midship are not near the values for a flat plate, i.e., Eq. 34.

Comparing the pressures along the same cuts, it is notable that the corresponding cuts give results of the same shape. However, the free-surface case produces a substantially smaller (more negative) pressure coefficient along the whole midship. The shear stresses and pressures for both cases converge towards the stern. Especially the results for the pressure distribution are influenced by the trapped air in the bottom region of the hull, causing heavy fluctuations along some lines.

As the double hull and free-surface cases are fundamentally different, they produce dissimilar flow regimes. As has been show in this work, clear differences both in the pressure and shear stress distributions on the hull can be found. This in turn leads to different values for both the friction, as well as the pressure resistances. Clearly, the ITTC-57 friction line for approximating ship resistance as a function of Reynolds number only is a gross simplification of the phenomenon. It would be desirable for future studies to focus on developing new and more sophisticated methods for ship resistance calculations.

The secondary objective of this work was to study the suitability of `OpenFOAM` for future work in the field of ship hydrodynamics. Evidently, the software can produce results of an acceptable level. However, the process of finding suitable run parameters among the myriad of choices offered by `OpenFOAM` turned out to be extremely cumbersome. Thus, an emphasis was put on robustness of the solution, possibly jeopardizing the accuracy of the results. Future studies could focus on finding the optimal discretization schemes and other numerical methods out of an accuracy standpoint. Also, it is possible that the run times for the simulations could be lowered with optimal choice of schemes and methods.

In order to achieve a higher level of accuracy, future topics of interest could also include abandoning the use of wall functions, or at least to decrease the y^+ values to 30. As Fig. 23 and 28 indicate, the velocity is not treated near the surface. By extending the analysis all the way to the wall, the velocity could be captured with more accuracy. This would in turn lead to a more accurate representation of the shear stresses. It is possible that the current shear stresses are not correct, as can be shown by the large difference compared to the flat plate results. One final source of inaccuracy that could be treated in future studies is the trapped air at the bottom region of the hull. This clearly leads to a confusing pressure distribution at the region in question.

References

[1] Thomson, W. On the Waves Produced by a Single Impulse in Water of Any Depth, or in a Dispersive Medium. *Proc. R. Soc. Lond*, 1887, Vol. 42, pp. 80–83.

[2] Resistance Committee. *Proceedings of the 8th ITTC.* International Towing Tank Conference, Madrid, Spain, 1957.

[3] Evans, J., H. Basic design concepts, *Journal of the American Society for Naval Engineers*, 1959, Vol. 71, No. 4, pp. 671-678.

[4] Lamb, T. *Ship Design and Construction*, Society of Naval Architects and Marine Engineers, 2003. ISBN 978-0-939773-40-4

[5] Gray, A., Cuneo, B., Vlahopoulos, N., Singer, D. *The Rapid Ship Design Environment - Multi-Disciplinary Optimization of a U.S. Navy Frigate*, ASNE Day 2013: Engineering America's Maritime Dominance, Society of Naval Architects and Marine Engineers, Arlington, VA., 21-22.2.2013.

[6] Ferziger, J. H., Perić , M. *Computational Methods For Fluid Dynamics* 3rd reviewed edn. Berlin, Springer-Verlag, 2002. ISBN 3-540-42074-6

[7] Bertram, V. *Practical ship hydrodynamics* Oxford, MA, Butterworth-Heinemann, 2000. ISBN 9780080514529

[8] White, F. M. *Viscous Fluid Flow* 3rd edn. New York, NY, McGraw-Hill, 2006. ISBN 007-124493-X

[9] Richardsson, L. F. *Weather Prediction by Numerical Process.* Cambridge, Cambridge University Press, 1922.

[10] Kolmogorov, A. N. The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. Translated to English in: *Proceedings: Mathematical and Physical Sciences*, 1991, Vol. 434, No. 1890, pp. 9-13.

[11] Rautaheimo, P. Developments in Turbulence Modelling with Reynolds-Averaged Navier-Stokes Equations. Doctoral Thesis, Helsinki University of Technology, Espoo, 2001.

[12] Pope, S. B. *Turbulent Flows* Cambridge, Cambridge University Press, 2000. ISBN 0-521-59125-2

[13] Spalart, P. R., Allmaras, S. R. A One-Equation Turbulence Model for Aerodynamic Flows. *Recherche Aerospatiale*, 1994, No. 1, pp. 5–21.

[14] Menter, F. R. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal*, 1994, Vol. 32, No. 8, pp. 1598–1605.

[15] Wilcox, C. W. Reassessment of the Scale-Determining Equation for Advanced Turbulence Models. *AIAA Journal*, 1988, Vol. 26, No. 11, pp. 1299–1310.

[16] Wilcox, C. W. Formulation of the $k - \omega$ Turbulence Model Revisited. *AIAA Journal*, 2008, Vol. 46, No. 11, pp. 2823–2838.

[17] Menter, F. R., Kuntz, M., Langtry, R. Ten Years of Industrial Experience with the SST Turbulence Model. In Hanjalic, K., Nagano, Y., Tummers, M, (eds.) *Turbulence, Heat and Mass Transfer 4*, Begell House Inc. 2003, pp. 625–632.

[18] Zhang, Zhi-rong. Verification and validation for RANS simulation of KCS container ship without/with propeller. *Journal of Hydrodynamics, Ser. B*, 2010, Vol. 22, No. 5, pp. 932–939.

[19] Phillips, W. R. C., Ratnanather, J. T. The outer region of a turbulent boundary layer. *Physics of Fluids*, 1990, Vol. 2, No. 3, pp. 427–434.

[20] Haase, M., Binns, J., Thomas, G., Bose, N. Resistance Prediction of Medium-speed Catamarans Using Free-surface Viscous Flow Simulations. 15th Numerical Towing Tank Symposium, 2012.

[21] Wilcox, D. C. *Turbulence modeling for CFD* La Cañada, CA, DCW Industries, 1993. ISBN 0-9636051-0-0

[22] Hirt, C. W., Nichols, B. D. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 1981, Vol. 39, No. 1, pp. 201–255.

[23] Chen, S., Johnson, D. B., Raad, P. E., Fadda, D. The surface marker and micro-cell method. *International Journal For Numerical Methods in Fluids*, 1997, Vol. 25, pp. 749–778.

[24] Darwish, M., Moukalled, F. Convective Schemes for Capturing Interfaces of Free-Surface Flows on Unstructured Grids. *Numerical Heat Transfer Part B*, 2006, Vol. 49, pp. 19–42.

[25] Ubbink, O. Numerical prediction of two fluid systems with sharp interfaces. Doctoral Thesis, Imperial College, London, 1997.

[26] Blasius, H. Grenzschichten in Flüssigkeiten mit kleiner Reibung. Translated to English in: *Technical Memorandium 1256*, National Advisory Committee for Aeronautics, 1950.

[27] Denli, N., Landweber, L. Thick axisymmetric turbulent boundary layer on a circular cylinder. *Journal of Hydronautics*, 1979, Vol. 13, No. 3, pp. 92–104.

[28] Larsson, M., Raven, H. C. *The Principles of Naval Architecture Series: Ship resistance and flow* Jersey City, NJ, Society of Naval Architects and Marine Engineers, 2010. ISBN 0939773767

[29] Wigley, W. C. S. A Comparison of Experiment and Calculated Wave-Profiles and Wave-Resistances for a Form Having Parabolic Waterlines. *Proc. R. Soc. Lond*, 1934, Vol. 144, pp. 144–163.

[30] Mitchell, J. H. The Wave Resistance of a Ship. *Phil. Mag.*, 1898, Vol. 45, pp. 106–123.

[31] Froude, W. Observations and suggestions on the subject of determining by experiment the resistance of ships. *The Papers of William Froude, 1810-1879*, London, UK: Royal Institution of Naval Architects, 1955.

[32] Hughes, G. Fricition and form resistance in turbulent flow and a proposed formulation for use in model and ship correlation. *Trans. RINA*, 1954, Vol. 96.

[33] Schoenherr, K. E. Resistance of Flat Surfaces Moving Through a Fluid. *Trans. SNAME*, 1932, Vol. 40.

[34] The OpenFOAM Foundation, *The OpenFOAM User Guide*. Retrieved 22.5.2013. Available at http://www.openfoam.org/docs/user/

[35] The OpenFOAM foundation, *OpenFOAM Programmer's Guide*, version 2.3.0 February 5th 2014. Retrieved 17.7.2014. Available at http://foam.sourceforge.net/docs/Guides-a4/ProgrammersGuide.pdf

[36] Patankar, S. V. *Numerical Heat Transfer and Fluid Flow* Taylor & Francis, 1980. ISBN 978-0-89116-522-4.

[37] Jasak, H. Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows. Doctoral Thesis, Department of Mechanical Engineering, Imperial College, London, 1996.

[38] Issa, R., I. Solution of the implicitly discretized fluid flow equations by operator-splitting *Journal of Computational Physics*, 1986, Vol. 62, pp. 40–65.

[39] Domingues, M. O., Gomes, S. M., Roussel, O., Schneider, K. An adaptive multiresolution scheme with local time stepping for evolutionary PDEs. *Journal of Computational Physics*, 2008, Vol. 227, No. 8 pp. 3758–3780.

[40] Weller, H. G. A new approach to VOF-based interface capturing methods for incompressible and compressible flows. Technical Report No. TR/HGW/04.

[41] van Leer, B. Towards the ultimate conservative difference scheme. V: A second-order sequel to Godunov's method. *Journal of Computational Physics*, 1979, Vol. 32, pp. 101–136.

[42] Siemens AG, *Parasolid: Siemens PLM software*. Retrieved 21.1.2014. Available at http://www.plm.automation.siemens.com/en_us/products/open/parasolid/index.shtml

[43] Esquivel de Pablo, P. Analysis of the Shear Stress Distributions on two Ship Hull Forms. Master's Thesis, Aalto University School of Engineering, Espoo, 2013.

[44] Eça, L., Hoekstra M. The Numerical Friction Line. *Journal of Marine Science and Technology*, 2008, Vol. 13, No. 4, pp. 328–345.

[45] ITTC 7.5-03-01-01. Uncertainty Analysis in CFD, Verification and Validation, Methodology and Procedures, Iternational Towing Tank Conference, 2008.

[46] Personal communications with STX Europe, 13.01.2014.

Appendix A Grid Generation

The mesh generation in HEXPRESS™starts by creating a domain. This part consists of building the initial geometry around the hull shape, whereafter the domain itself is created. This part should be done with scripts, they are easy to write, save time, and limit the amount of mistakes done. Below, a sample script for the generation of the domain is given:

```
#first the parasolid (or other) geometry is imported
HXP.import_parasolid("/hull.x_t")

#here the box around the hull is created
#set appropriate dimensions of the box
HXP.create_cube("HULLBOX",Point(-40.0,0.0,0.0),Point(20.0,20.0,10.0))
HXP.create_cube("UWATER",Point(-40.0,0.0,-10.0),Point(20.0,20.0,0.0))
HXP.unite_bodies("HULLBOX",["UWATER"])
# Remove existing domain
HXP.create_cube("ANTIHULL",Point(-1,-1,-0.5),Point(10,1,0.5))
HXP.substract_bodies("HULLBOX",["ANTIHULL"])

#front box
HXP.create_cube("B2",Point(20,0,-10),Point(30,20,10))
HXP.unite_bodies("HULLBOX",["B2"])

#the domain itself is created here
#the "hexpress general advice" explains the meaning of the parameters
HXP.create_domain("/testAppendix.dom",["HULLBOX"],0.009,0.35,0.0001,1,0.0001,1)

#the domain is imported to the next phase
HXP.import_domain("/hull.dom")
HXP.set_mesh_generation_mode("3D")
```

The second step of the mesh generation is the manipulation of the recently generated domain. In this phase the faces were first merged, so that each patch consist of only one face. After this the edges were merged in a similar fashion. Hereafter, the patches were given names and boundary conditions. HEXPRESS™does not recognize the same boundary conditions as `OpenFOAM`, so here "mirror" or "solid" were chosen for the corresponding patches and all other patches were set as "external".

Once the work on the domain had been completed, the generation of the mesh could start. In HEXPRESS™this was a five step process consisting of the following steps:

- **Initial mesh**, here the background mesh was created by dividing the cartesian axes into a desired number of parts.

- **Adapt to geometry**, in this step the mesh was refined around the hull ans well as the free surface. The amount of refinement levels depends on the background mesh and the desired cell size of the final mesh, so this step could be considered iterative. For the hull, four refinements were used while the target cells size was $0.0m$ in every direction.

  As the free surface was not defined by a geometry, the refinement for it was done through refinement boxes, defined by the following script:

```
HXP.delete_all_refinement_boxes()

#Box for the refinements in z-direction
HXP.create_refinement_cube(-26.5,0.0,-0.08, 30.0,20.0,0.08)
HXP.refinement_box(0).set_adaptation_flags(1,1)
HXP.refinement_box(0).set_target_size(1.0,1.0,0.0)
HXP.refinement_box(0).set_refinement_level(5)
HXP.refinement_box(0).set_diffusion_depth(0)

#Box for the refinements in the x- and y-directions
HXP.create_refinement_cube(-17.6,0.0,-0.08, 13.0,8.8,0.08)
HXP.refinement_box(1).set_adaptation_flags(1,1)
HXP.refinement_box(1).set_target_size(0.0,0.0,0.0)
HXP.refinement_box(1).set_refinement_level(3)
HXP.refinement_box(1).set_diffusion_depth(0)
```

- **Snap to geometry**, no user action is needed by default. However, problems with the viscous layers at a later stages were handled by using a forced Type II buffer insertion. This was done with the "set_typeII_on_mirror_faces plugin" found in the Marine package of plugins. By doing this, the viscous layers were guaranteed to extend around the hull all the way to the mirror-plane.

- **Optimize**, here all the default settings were used.

- **Viscous layers** is the final step of the mesh generation. It creates a refinement around the desired geometries, using three parameters: first layer thickness, growth ratio of cell height and number of cells in the refinement. Values $0.001m, 1.1, 15$ were used for the respective parameters. The number of cells parameter is limited by HEXPRESS™, so the refinement will stop once the inflation of layers will reach the mesh generated earlier.

Once the mesh had been finished, the last step was to export it from HEXPRESS™in `OpenFOAM` format. However, HEXPRESS™does not generate a working "boundary" file for `OpenFOAM`. It does not recognize the "symmetryPlane" patch used in `OpenFOAM` to define symmetry planes, and simply writes the symmetry planes as patches. This was corrected manually by replacing the corresponding type "patch" with type "symmetryPlane".

In order to ensure a better quality of the refinements around the free surface, the inlet was not divided into a water part and an air part in HEXPRESS™. Instead, this was done with the `topoSet` and `createPatch` utilities in `OpenFOAM`. The `topoSet` utility will pick user defined surfaces from the mesh, and the `createPatch` utility will create new patches out of these selected surfaces. The `topoSetDict` and `createPatchDict` used are given below.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      topoSetDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
actions
(
    {
        name    inlet_air;
        type    faceSet;
        action  new;
        source  boxToFace;
        sourceInfo
        {
            box (29.98 0.0 0.0)(30.02 20.0 10.0);
        }
    }

    {
        name    inlet_water;
        type    faceSet;
        action  new;
        source  boxToFace;
        sourceInfo
        {
            box (29.98 0.0 -10.0)(30.02 20.0 0.0);
        }
    }
);
// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      createPatchDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
pointSync false;
patches
(
    {
        // Name of new patch
        name inlet_air;

        // Dictionary to construct new patch from
        patchInfo
        {
            type patch;
            neighbourPatch inlet_water;
        }

        // How to construct: either from 'patches' or 'set'
        constructFrom set;

        // If constructFrom = set : name of faceSet
        set inlet_air;
    }

   {     name inlet_water;

        patchInfo
        {
            type patch;
            neighbourPatch inlet_air;
        }

        constructFrom set;
        set inlet_water;

        }

  );

// ************************************************************************* //
```

Appendix B Sample Boundary Conditions

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alpha;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 0 0 0 0 0 0];
internalField   uniform 0;
boundaryField
{
    inlet_water
    {
        type            fixedValue;
        value           uniform 1;
    }
    inlet_air
    {
        type            fixedValue;
        value           uniform 0;
    }
    outlet
    {
        type            zeroGradient;
    }
    side
    {
        type            zeroGradient;
    }
    top
    {
        type            zeroGradient;
    }
    top_front
    {
        type            zeroGradient;
    }
    bottom
    {
        type            zeroGradient;
    }
    bottom_front
    {
        type            zeroGradient;
    }
    hull
    {
        type            zeroGradient;
    }
    mirror
    {
        type            symmetryPlane;
    }
}
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
turbulentK      0.00944178;
dimensions      [0 2 -2 0 0 0 0];
internalField   uniform $turbulentK;
boundaryField
{
    inlet_water
    {
        type            fixedValue;
        value           uniform $turbulentK;
    }
    inlet_air
    {
        type            fixedValue;
        value           uniform $turbulentK;
    }
    outlet
    {
        type            zeroGradient;
    }
    side
    {
        type            zeroGradient;
    }
    top
    {
        type            zeroGradient;
    }
    top_front
    {
        type            zeroGradient;
    }
    bottom
    {
        type            zeroGradient;
    }
    bottom_front
    {
        type            zeroGradient;
    }
    hull
    {
        type            kqRWallFunction;
        value           uniform 0;
    }
    mirror
    {
        type            symmetryPlane;
    }
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      omega;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
turbulentOmega  2.564253;
dimensions      [0 0 -1 0 0 0 0];
internalField   uniform $turbulentOmega;
boundaryField
{
    inlet_air
    {
        type            fixedValue;
        value           uniform $turbulentOmega;
    }
    inlet_water
    {
        type            fixedValue;
        value           uniform $turbulentOmega;
    }
    outlet
    {
        type            zeroGradient;
    }
    side
    {
        type            zeroGradient;
    }
    top_front
    {
        type            zeroGradient;
    }
    top
    {
        type            zeroGradient;
    }
    bottom
    {
        type            zeroGradient;
    }
    bottom_front
    {
        type            zeroGradient;
    }
    hull
    {
        type            omegaWallFunction;
        value           uniform $turbulentOmega;
    }
    mirror
    {
        type            symmetryPlane;
    }
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [1 -1 -2 0 0 0 0];
internalField   uniform 0;
boundaryField
{
    inlet_water
    {
        type            zeroGradient;
    }
    inlet_air
    {
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    side
    {
        type            zeroGradient;
    }
    top
    {
        type            zeroGradient;
    }
    top_front
    {
      type              fixedValue;
      value             uniform 0;
    }
    bottom
    {
        type            zeroGradient;
        value           uniform 0;
    }
    bottom_front
    {
      type              fixedValue;
      value             uniform 0;
    }
    hull
    {
        type            zeroGradient;

    }
    mirror
    {
        type            symmetryPlane;
    }
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
velU            -2.2668;
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform ($velU 0 0);
boundaryField
{
    inlet_water
    {
        type            fixedValue;
        value           uniform ($velU 0 0);
    }
    inlet_air
    {
        type            fixedValue;
        value           uniform ($velU 0 0);
    }
    outlet
    {
        type            zeroGradient;
    }
    side
    {
        type            fixedValue;
        value           uniform ($velU 0 0);
    }
    top
    {
        type            fixedValue;
        value           uniform ($velU 0 0);
    }
    top_front
    {
        type            fixedValue;
        value           uniform ($velU 0 0);
    }
    bottom
    {
        type            fixedValue;
        value           uniform ($velU 0 0);
    }
    bottom_front
    {
        type            fixedValue;
        value           uniform ($velU 0 0);
    }
    hull
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    mirror
    {
        type            symmetryPlane;
```

Appendix C Sample System Files

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
application     LTSInterFoam;
startFrom       latestTime;
startTime       0.0;
stopAt          endTime;
endTime         30000.0;
deltaT          1.0;
writeControl    runTime;
writeInterval   5000.0;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression compressed;
timeFormat      general;
timePrecision   6;
runTimeModifiable yes;

functions
{
    forces
    {
        type forces;
        functionObjectLibs ("libforces.so");
        patches ("hull*");
        rhoName rho;
        rhoInf 1000;
        CofR (0 0 0);
        outputControl timeStep;
        outputInterval 1;
        log    true;
    }
    forceCoeffs
    {
        type forceCoeffs;
        functionObjectLibs ("libforces.so");
        patches ("hull*");
        rhoName rho;
        rhoInf 1000;
        CofR (0 0 0);
        dragDir (-1 0 0);
        liftDir (0 0 1);
        pitchAxis (0 1 0);
        outputControl timeStep;
        outputInterval 1;
        log    true;
        magUInf   2.2668;
        lRef      8.84;
        Aref      7.5748355;
    }
}
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 32;

    method             scotch;

distributed     no;

roots           ( );
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
defaultFieldValues
(
    volScalarFieldValue alpha1 0
);

regions
(
    boxToCell
    {
        box (-100 -100 -100) (100 100 0);
        fieldValues
        (
            volScalarFieldValue alpha1 1
        );
    }
);
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
solvers
{
    pcorr
    {
        solver          PCG;
        preconditioner
        {
            preconditioner  GAMG;
            smoother        DICGaussSeidel;
            agglomerator    faceAreaPair;
            mergeLevels     1;
            nCellsInCoarsestLevel 100;
            cacheAgglomeration true;
            tolerance       1e-5;
            relTol          0;
        };

        tolerance       1e-5;
        relTol          0;
    };

    p_rgh
    {
        solver          GAMG;

        smoother        GaussSeidel;
        agglomerator    faceAreaPair;
        mergeLevels     1;
        nCellsInCoarsestLevel 100;
        cacheAgglomeration true;
        nPreSweeps      0;
        nPostSweeps     0;
        tolerance       1e-6;
        relTol          0.01;
    };
```

```
    p_rghFinal
    {
        $p_rgh;
        tolerance       1e-7;
        relTol          0;
    }

    "(U|k|omega).*"
    {
        solver          PBiCG;
        preconditioner  DILU;
        smoother        GaussSeidel;
        nSweeps         1;
        tolerance       1e-7;
        relTol          0.0;
    };
}

PIMPLE
{
    momentumPredictor yes;
    nCorrectors     1;
    nNonOrthogonalCorrectors 1;
    nAlphaCorr      1;
    nAlphaSubCycles 2;
    cAlpha          1;
    maxCo           0.5;
    maxAlphaCo      0.2;
    nAlphaSweepIter 1;
    rDeltaTSmoothingCoeff 0.1;
    rDeltaTDampingCoeff 1;
    maxDeltaT       1;
}

relaxationFactors
{
    fields
    {
    p               0.3;
    }
    equations
    {
     U              0.7;
     k              0.7;
     epsilon        0.7;
     R              0.7;
     nuTilda        0.7;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
ddtSchemes
{
    default          localEuler rDeltaT;
}
gradSchemes
{
    default cellMDLimited Gauss linear 0.5;
}
divSchemes
{
    div(rho*phi,U)      Gauss linearUpwind grad(U);
    div(phi,alpha)      Gauss vanLeer;
    div(phirb,alpha)     Gauss interfaceCompression;
    div(phi,k)            Gauss linearUpwind grad(U);
    div(phi,omega)      Gauss linearUpwind grad(U);
    div((muEff*dev(T(grad(U))))) Gauss linear;
}
laplacianSchemes
{
    default         Gauss linear limited 0.5;

}
interpolationSchemes
{
    default         linear;
}
snGradSchemes
{
    default         limited 0.5;
}
fluxRequired
{
    default         no;
    p_rgh;
    pcorr;
    alpha1;
}

// ************************************************************************* //
```

Appendix D Sample Constant Files

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           (0 0 -9.81);

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
RASModel        kOmegaSST;

turbulence      on;

printCoeffs     on;
// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
simulationType  RASModel;
// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
phase1
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ]  1e-06;
    rho             rho [ 1 -3 0 0 0 0 0 ] 1000;
}
phase2
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ]  1.48e-05;
    rho             rho [ 1 -3 0 0 0 0 0 ] 1;
}

sigma           sigma [ 1 0 -2 0 0 0 0 ] 0;
// ************************************************************************* //
```